MASTER EN SCIENCES INFORMATIQUES

Prototypage du processus de conception de la filière mécanique d'un véhicule automobile

De Wasseige, Olivier

Award date:
1986

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES
NOTRE-DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

PROTOTYPAGE DU PROCESSUS DE CONCEPTION DE LA FILIERE MECANIQUE D'UN VEHICULE AUTOMOBILE

OLIVIER DE WASSEIGE

ANNEE ACADEMIQUE 1985-1986

PROMOTEUR : MONSIEUR FRANCOIS BODART

MEMOIRE PRESENTE EN VUE
DE L'OBTENTION DU GRADE DE
LICENCIE ET MAITRE
EN INFORMATIQUE

REMERCIEMENTS

Mes premiers remerciements vont à Monsieur François BODART, Professeur à l'Institut d'Informatique de Namur.
Non seulement il a, en tant que directeur de ce mémoire, organisé le stage de préparation, suivi la résolution du problème, orienté la rédaction, mais il m'a aussi appris beaucoup sur les plans professionnel et surtout humain.

Ce mémoire n'aurait pas été possible sans l'apport de six mois de stage passés en France, au service de Systèmes d'Informations Techniques de la Régie Renault.
A cet égard, qu'il me soit permis de remercier Madame Elaine PENNY, responsable du département Méthodologies et Madame Nadine LECLAIR, chef du projet Filière Méthodes Mécaniques, pour l'intérêt qu'elles ont porté au sujet du mémoire et pour le contexte favorable et constructif qu'elles ont créé pendant ces six mois à Paris.
Lors de ce stage, la compréhension de l'organisation de la Régie et des objectifs du projet Filière Méthodes Mécaniques, l'analyse du problème à résoudre et l'implémentation d'une solution n'auraient pu se concevoir sans l'aide précieuse, journalière et compétente de Monsieur Erik GHESQUIERE, à qui je dois beaucoup et que je remercie infiniment.

Je n'oublie pas Mesdames Anne-Marie HENNEBERT, Brigitte JADOT et Pascale DRIANNE, et Monsieur Jean-Marie LEHEUREUX, avec lesquels la collaboration dans la résolution du problème par le logiciel IDA développé à l'Institut d'Informatique a été très fructueuse.

Un mémoire représente l'aboutissement de nombreuses années d'études et par conséquent de cours donnés par des professeurs que je tiens à remercier tous pour leur compétence et leur contact humain.

Enfin, un grand merci à mes parents qui m'ont permis de poursuivre des études, à mes proches qui m'ont soutenu dans ce travail de longue haleine, et à tous ceux que j'aurais oubliés.

Olivier de Wasseige

PLAN

PLAN

PROTOTYPAGE DU PROCESSUS DE CONCEPTION DE LA FILIERE MECANIQUE D'UN VEHICULE AUTOMOBILE

INTRODUCTION GENERALE

Ce mémoire a plusieurs objectifs.

Le premier objectif est de présenter le contexte général de réalisation du mémoire.
La première partie présente à cette fin le système d'information, support du processus de conception de la filière mécanique d'un véhicule automobile. Ce système d'information a été étudié au cours d'un stage à la Direction des Services de Conception et fabrication assistée par ordinateur et d'Informations Techniques (D.S.C.I.T.) de la Régie Nationale des Usines Renault (R.N.U.R.), où est développé le projet Filière Méthodes Mécaniques à destination de la Direction des Méthodes Centrales (D.M.C.).
Après une description générale du processus de conception de la filière mécanique d'un véhicule automobile, nous présentons le problème à résoudre.

Le second objectif est de décrire la solution développée. Celle-ci concerne l'application "Plans de pièces" qui est une des fonctions réalisées par la Gestion Prévisionnelle, sous-système de la D.M.C.
A cette fin, la seconde partie présente le schéma conceptuel de cette application. Nous trouvons d'abord le schéma conceptuel des données, ensuite sa justification méthodologique, puis nous présentons le schéma conceptuel des traitements.

Le troisième objectif est la mise en oeuvre de cette application à l'aide de techniques de prototypage rapide.
La troisième partie présente d'abord différents aspects du prototypage de l'application "plans de pièces".
Ensuite, nous présentons les deux outils de prototypage utilisés: ORACLE, système de gestion de bases de données relationnelles doté d'un langage de quatrième génération, et DSL-PROTO, outil de prototypage développé dans le contexte de l'atelier-logiciel IDA (Interactive Design Approach) (Institut d'Informatique - Namur).
Nous expliquons la mise en oeuvre de cette application à l'aide des deux logiciels, en commentant les caractéristiques de réalisation de l'application en fonction du logiciel.
Nous terminons par la comparaison des deux outils dans le cadre du prototypage de l'application "Plans de pièces".

Le choix de rédaction porte sur la mise en oeuvre des aspects méthodologiques, d'une part liés à la modélisation du problème, d'autre part liés à la mise en oeuvre des outils, et ce plutôt que sur le contenu de la réalisation.
On trouve en annexe de larges descriptions de la réalisation.

Il est clair que le lecteur est supposé être familier avec les notions de l'analyse fonctionnelle (on consultera à ce sujet [BOD-83]). De plus, le lecteur doit avoir de sérieuses connaissances du logiciel IDA.
Quant au logiciel ORACLE, il est préférable que le lecteur en connaisse les caractéristiques principales, et qu'il soit familier au langage SQL.

PREMIERE PARTIE : LE SYSTEME D'INFORMATION, SUPPORT DU PROCESSUS DE CONCEPTION DE LA FILIERE MECANIQUE D'UN VEHICULE AUTOMOBILE

INTRODUCTION

Le but de la première partie est de décrire le système d'information servant de support au processus de conception de la filière mécanique d'un véhicule automobile. Il s'agit d'une partie introductive au travail réalisé.
Dans un premier chapitre, nous décrivons de façon générale le processus de conception de la filière mécanique d'un véhicule automobile.
Après avoir situé la filière mécanique dans l'ensemble du processus de conception, nous détaillons le projet Filière Méthodes Mécaniques.
Nous terminons par une description des sous-systèmes de la filière Méthodes Mécaniques.
Le second chapitre décrit le problème à résoudre : nous passons en revue les objectifs informationnels, deux principes de base, puis nous décrivons l'approche retenue en termes de système d'information.

CHAPITRE 1 : DESCRIPTION GENERALE DU PROCESSUS DE CONCEPTION DE LA FILIERE MECANIQUE D'UN VEHICULE AUTOMOBILE

INTRODUCTION

Le but de ce premier chapitre est d'introduire brièvement le processus de conception d'un véhicule automobile, principalement en ce qui concerne la filière mécanique de celui-ci.

Nous présentons aussi un résumé succint du projet informatique dénommé Projet "Filière Méthodes Mécaniques", développé au sein du département Systèmes d'Informations Techniques (S.I.T.) de la Direction des Systèmes de Conception et fabrication assistée par ordinateur et d'Informations Techniques (D.S.C.I.T.) de la Régie Nationale des Usines Renault (R.N.U.R.).

A cette fin, le chapitre 1 décrit d'abord la situation de la filière Méthodes Mécaniques au sein du processus de conception d'un véhicule automobile. Ensuite, les points importants du projet "Filière Méthodes Mécaniques" sont abordés. Enfin, on donne une description plus précise des différents aspects et domaines couverts par la Filière Méthodes Mécaniques.

1.1. SITUATION DE LA FILIERE MECANIQUE

1.1.1. VUE LINEAIRE DU PROCESSUS DE CONCEPTION

Le processus de conception et de réalisation d'un véhicule automobile est composé de trois grandes étapes :

1. La conception du véhicule en Bureau d'Etudes.
2. L'industrialisation et la maintenance des processus et des moyens aux départements des Méthodes.
3. La fabrication en usines.

Ces trois étapes ne sont pas purement séquentielles. Des mécanismes de "feed-back", de retour en arrière existent entre les étapes 1 et 2, 2 et 3, 1 et 3.

Les Méthodes ayant pour mission la définition du mode et des moyens de fabrication, ainsi que la réalisation, la mise en route et la maintenance de ceux-ci, sont amenées à remettre en question les projets et les plans du Bureau d'Etudes, en fonction des opportunités et des coûts des moyens de fabrication, de la faisa-

bilité technique des projets, etc... Le Bureau d'Etudes révise donc continuellement les plans de conception en fonction du travail des Méthodes.
C'est le seul "feed-back" que nous retiendrons.

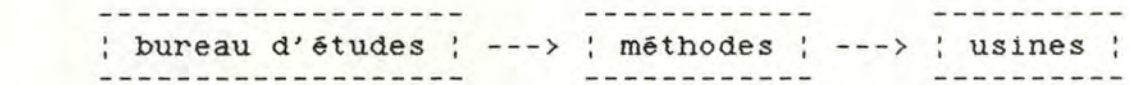


figure 1.1. : vue linéaire du processus de conception

1.1.2. VUE MATRICIELLE DU PROCESSUS DE CONCEPTION

Le processus de conception s'applique aux trois parties essentielles d'un véhicule automobile :
1. la filière électrique.
2. la filière carrosserie.
3. la filière mécanique .

Une filière correspond à un ensemble de données homogènes et pour lequel le flux vers l'extérieur est minimal.
On peut donc représenter le processus de conception d'un véhicule automobile de façon matricielle, l'axe horizontal figurant l'espace et l'axe vertical le temps.

ETAPE	FILIERE		
	ELECTRIQUE	CARROSSERIE	MECANIQUE
CONCEPTION EN BUREAU D'ETUDES			
INDUSTRIALISATION DES PROCESSUS ET MOYENS			METHODES MECANIQUES
FABRICATION EN USINES			

figure 1.2. : vue matricielle du processus de conception

1.2. LE "PROJET FILIERE METHODES MECANIQUES"

1.2.1. CHAMP D'APPLICATION

Si l'on se réfère à la figure 1.2. (point 1.1.2.), on constate que les Méthodes Mécaniques se situent à l'intersection de l'étape d'industrialisation des processus et des moyens et de la filière mécanique.
Il s'agit de la partie à laquelle s'applique le projet développé par la D.S.C.I.T.
On appelle donc, par abus de langage, Filière Méthodes Mécaniques ce qui correspond en fait à la partie Méthodes Mécaniques.
Nous conserverons dès à présent la terminologie Filière Méthodes Mécaniques pour respecter les termes employés dans les documents relatifs au projet.

Pour plus de précision, il faut souligner que les Méthodes sont divisées en deux composants : Méthodes Centrales et Méthodes Décentralisées. Le Projet Filière Méthodes Mécaniques s'applique aux Méthodes Centrales Mécaniques, et à leur demande, aux Méthodes Décentralisées Mécaniques, à savoir les filiales mécaniques, les départements intégrés, etc...

Les Méthodes Centrales sont aussi appelées Direction des Méthodes Centrales (D.M.C.) et on appelle le Bureau d'Etudes aussi Direction de la Planification et des Investissements (D.P.I.)

1.2.2. LES SIX CAUSES PROFONDES D'INSATISFACTION

Les causes profondes d'insatisfaction " amènent les responsables d'une organisation à envisager la modification de la situation existante du Système d'Information pour en améliorer les performances et les qualités par la mise en oeuvre éventuelle de technologies informatiques nouvelles. " [BOD-83]
Nous allons passer en revue les causes d'insatisfaction décelées.

1) Prix de revient de fabrication

Le prix de revient de fabrication est trop élevé. Deux des causes principales sont le temps de réaction qui est trop élevé et la qualité des solutions techniques qui est améliorable.

2) Circulation de l'information

La circulation de l'information au sein des Méthodes Mécaniques est mauvaise. Les documents sont nombreux, divers, variés, reprenant sous des formes multiples tout ou partie des mêmes informations.

3) Formalisation des connaissances

Les connaissances, principalement le savoir-faire des agents des Méthodes, sont mal formalisées.

4) Diffusion des connaissances et de la documentation technique

La diffusion des connaissances et de la documentation technique est rendue difficile par la variété de documents générés.
De plus, la non-continuité de la diffusion de l'information ne donne pas aux agents des Méthodes l'assurance de posséder la bonne information au bon moment.

5) Difficultés d'analyse

Les difficultés d'analyse de l'existant entrainent l'obligation de générer des nouveaux documents, non officiels, contenant les informations nécessaires à ce type d'activité, informations récoltables au prix d'une grande somme de travail de recherche.

6) Déphasage entre chronologie du travail et chronologie de prise de responsabilités

Les moyens informatiques actuels sont déphasés par rapport à la réalité du travail effectué. Conçus sur la répartition des responsabilités et sur leur échéancier, ils ne reflètent pas la réalité des relations logiques existant entre les différents acteurs.
Il y a donc déphasage entre chronologie du travail et chronologie de prise de responsabilités, l'ensemble des systèmes informatiques existants s'appuyant sur la chronologie de prise de responsabilités.

1.2.3. OBJECTIFS DE L'ORGANISATION

Les objectifs de l'organisation " résultent de l'analyse des causes profondes d'insatisfaction ... Ces objectifs concernent les comportements opérationnels ou les comportements de gestion de l'organisation. " [BOD-83]

Voyons quels éventuels objectifs de l'organisation correspondent aux causes d'insatisfaction décrites au point 1.2.2.

* Prix de revient de fabrication

Pour diminer le prix de revient de fabrication, on améliorera les solutions techniques grâce à l'établissement de différentes hypothèses technologiques, par un système d'aide à l'étude Méthodes. Quant au temps de réaction, on diminuera les coûts de mise en route par l'emploi de meilleures solutions techniques et par le rapprochement le plus fin possible entre la théorie et la réalité de l'existant.

* Formalisation des connaissances

L'objectif sera de permettre une certaine formalisation du savoir-faire des agents Méthodes dans un but de capitalisation et de réutilisation de celui-ci, ce qui entraînera une amélioration de la qualité des solutions choisies.

* Diffusion des connaissances et de la documentation technique

L'objectif serait la mise à disposition et la diffusion des connaissances au plus grand nombre nécessaire et suffisant d'agents des Méthodes.

* Déphasage entre chronologie du travail et chronologie de prise de responsabilités

L'objectif est de posséder des outils informatiques pour le "travail réel" et non des outils s'appuyant sur les "structures chronologiques" de prise de responsabilités.

1.3. DESCRIPTION DE LA FILIERE METHODES MECANIQUES

1.3.1. PRESENTATION DES ACTEURS DE LA FILIERE

Nous avons vu comment situer la Filière Méthodes Mécaniques dans le cadre du processus de conception d'un véhicule automobile.
Nous avons dit que la mission de la Filière Méthodes Mécaniques était la définition du mode et des moyens de fabrication, ainsi que la réalisation, la mise en oeuvre et la maintenance de ceux-ci.

Détaillons l'organisation et les activités de la Filière Méthodes Mécaniques.
L'activité des Méthodes Mécaniques comprend trois grands secteurs: la coordination, la préparation et les services techniques. Chaque secteur est composé de services.

Poste de travail	Secteur	Service
FILIERE METHODES MECANIQUES	COORDINATION (GESTION PREVISIONNELLE)	COORDINATION
		PERSONNEL
	PREPARATION (DOCUMENTATION TECHNIQUE)	PREPARATION MOTEUR
		PREPARATION BOITE
		PREPARATION CHASSIS
	SERVICES TECHNIQUES (CONCEPTION DES MOYENS)	MACHINE
		CONTROLE OUTILS COUPANTS
		TRT THERMIQUE - LAVAGE
		INSTALLATIONS

figure 1.3. : les activités des Méthodes Mécaniques

Cette classification ne tient pas compte de l'éventuelle centralisation ou décentralisation des services, point qui nous intéresse moins.

Examinons les rôles des trois secteurs. Ces rôles sont généralement valables pour tous les services d'un secteur.

1. **COORDINATION** :
   - analyser les plans de fabrication proposés par le Bureau d'Etudes.
   - définir en liaison avec la préparation la manière de rendre les moyens plus capacitaires.
   - fournir au contrôle de gestion les investissements industriels et prévisionnels.
   - établir des dossiers pour l'ensemble des instances de décisions de l'entreprise.

2. **PREPARATION** :
   - elle est chargée de faire évoluer le projet du stade prévisionnel au stade industriel.
   - elle établit l'ensemble des documents qui permettront à la Fabrication d'industrialiser le projet.
   - pour parvenir à ces buts, la préparation est chargée de quatre grands types d'activité:
     * établissement de la gamme de fabrication
     * coordination des actions
     * responsabilité des actions de diffusion
     * conception de l'implantation.

3. **SERVICES TECHNIQUES** : à partir des demandes des secteurs de préparation :
   - définissent les moyens de fabrication.
   - préconisent pour l'entreprise, après essais et tests, les conditions d'utilisation des biens d'équipement et des machines.
   - assurent la mise en route sur site des moyens.
   - étudient de nouveaux moyens de production.
   - orientent le Bureau d'Etudes vers de nouvelles solutions techniques.

" De l'observation de la Filière Méthodes Mécaniques, on met en évidence les quatre sous-systèmes fonctionnels suivants :

* la Gestion Prévisionnelle
* la Documentation Technique
* la Conception des Moyens
* la Gestion de Projet.

Aux trois premiers sous-systèmes correspond la typologie des acteurs de la filière :

* les coordinateurs
* les préparateurs
* les techniciens.

Le quatrième, la Gestion de Projet, s'appuie sur les trois précédents. (N.D.L.R. : i.e. gère la juxtaposition et les relations des trois précédents). Son objectif est de planifier et coordonner au mieux les activités des Méthodes Mécaniques et de contrôler les impacts de modifications et les coûts associés.

A ces quatre sous-systèmes, on peut ajouter les sous-systèmes techniques suivants :

* la Documentation des Connaissances (la capitalisation du savoir-faire et sa réutilisation de manière ergonomique).
* les Moyens Techniques et Utilitaires (interfaces, outils CFAO, communication, sécurité, confidentialité, éditions, contrôle, exploitation ...). En quelque sorte une boîte à outils utile pour tous les sous-systèmes fonctionnels précédents. " [REN-1-85]

Nous détaillons ci-dessous les activités des trois sous-systèmes fonctionnels les plus importants, à savoir Gestion Prévisionnelle, Documentation Technique et Conception des Moyens.

1.3.2. LA GESTION PREVISIONNELLE (COORDINATION)

La Gestion Prévisionnelle consiste en l'exploitation et le chiffrage économique des plans de fabrication pour préparer les dossiers de décision. Elle recouvre les travaux de coordination. On y retrouve principalement :

A) Etablissement du plan de pièces

- établissement de nomenclatures de pièces pour réaliser des hypothèses de constitution des véhicules futurs et prendre en compte les plans du Bureau d'Etudes.
- traduction des besoins de véhicules en besoins d'organes et de pièces.
- répartition de la fabrication et du montage des pièces et organes sur les différents lieux de production.

B) Chiffrage, valorisation économique

" Pour, par site de production, compte tenu des volumes, des moyens de production existants, des hypothèses de fabrication relatives aux pièces ou organes à produire, des ratios de coûts des différentes fonctions méthodes à réaliser sur ces pièces ou organes, chiffrer les coûts élémentaires et réaliser les synthèses nécessaires. " [REN-1-85]

C) Gestion du poste de travail

"- pour faire en sorte qu'une hypothèse qui doit être officialisée puisse être accueillie au niveau du système central.
- pour permettre à l'agent Méthodes de mettre en place les structures d'information personnelles et nécessaires à l'élaboration de son travail.
- pour gérer les différentes étapes du travail, assurer des retours en arrière faciles, diversifier les hypothèses sans refaire toute la démarche". [REN-1-85]

1.3.3. LA DOCUMENTATION TECHNIQUE (PREPARATION)

On élabore la Documentation Technique relative au processus de fabrication. Cela recouvre les travaux de préparation des moteurs, des boîtes de vitesses et des châssis.
On y retrouve principalement :

A) Conception des gammes

- conception des fiches techniques concernant les gammes de fabrication.
- choix des types de machines adaptés aux besoins exprimés au travers des fiches techniques.

B) Conception des implantations

"- pour étudier, simuler, visualiser l'implantation optimale des moyens concernés par une ou plusieurs gammes de fabrication en un site géographique donné.
- pour comparer les différentes hypothèses.
- pour établir et ensuite actualiser par la prise en compte des informations en provenance de la production des cas de fonctionnement des machines". [REN-1-85]

C) Gestion du poste de travail

Mêmes traitements que pour la gestion prévisionnelle (cfr point 1.3.2. C)).

1.3.4. LA CONCEPTION DES MOYENS (SERVICES TECHNIQUES)

Nous décrivons ici la conception des moyens et biens d'équipements nécessaires à la fabrication ou au montage des pièces dont le processus a été établi à l'aide du sous-système "Documentation Technique" (cfr point 1.3.3.). Ce sous-système recouvre les travaux des services techniques.
On y retrouve principalement :

A) Conception des outils et des machines

- choix à l'aide des informations enregistrées au niveau du sous-système de documentation des connaissances des conditions optimales d'utilisation des moyens compte tenu des cadences et du processus exprimé au niveau des opérations de la gamme.
- simulation des conditions de fonctionnement.
- production des plans et nomenclatures des machines et outils conçus.

B) Etablissement des standards d'outillages et de machines

pour constituer les bibliothèques graphiques.

C) Gestion du poste de travail

Mêmes traitements que pour la gestion prévisionnelle (cfr point 1.3.2. C)).

CHAPITRE 2 : LE PROBLEME A RESOUDRE

INTRODUCTION

Nous avons décrit de façon générale, dans le premier chapitre, le processus de conception de la filière mécanique d'un véhicule automobile. Nous détaillons, dans le deuxième chapitre, le problème à résoudre.

A cette fin, nous spécifions d'abord les objectifs informationnels. Ensuite, nous donnons deux principes de base permettant de réaliser les objectifs spécifiés.
Enfin, nous décrivons l'approche retenue en terme de système d'information.
Nous énumérons et expliquons les trois grandes finalités de la base de données à construire : la mémorisation, la vision intégrée des données et les tests des hypothèses.

2.1. SPECIFICATION DES OBJECTIFS INFORMATIONNELS

Il est intéressant de spécifier les objectifs informationnels, afin de bien voir les éléments du problèmeà résoudre.

D'après [BOD-83], les objectifs informationnels sont déduits des objectifs de l'organisation (cfr point 1.2.3.).
Ces objectifs à assigner au comportement du S.I. permettront d'atteindre les objectifs de l'organisation.

Il y a deux catégories d'objectifs informationnels :
- ceux exprimant les qualités de l'information produite par les processeurs du S.I.
- ceux caractérisant le comportement des processeurs du S.I.

A. Objectifs informationnels exprimant les qualités de l'information.

Les objectifs informationnels exprimant les qualités de l'information sont :
- améliorer la fiabilité et la qualité de la documentation technique.

- ce premier objectif englobe les objectifs suivants :
* amélioration de la qualité des solutions techniques.

* formalisation du savoir-faire, qui existe, mais n'apas encore été formalisé.

- mieux localiser l'information : * dans l'espace : où se trouve l'information ?
  * dans le projet : comment se situe-t'elle dans le projet?

- du point de vue de la pertinence : avoir unevue des données communes propres au poste de travail.

- améliorer la confidentialité.

B. Objectifs informationnels caractérisant le comportement des processeurs d'un S. I.

Du point de vue des performances, les objectifs informationnels caractérisant le comportement des processeurs d'un S. I. sont :
- améliorer les temps de réaction de la Filière Méthodes Mécaniques (ce qui permettra d'améliorer la concertation et la coordination entre les parties amont et aval).
- répondre plus vite aux faits et aux hypothèses projetés au niveau technique.
Cela permettra de diminuer le prix de revient de fabrication.

2.2. PRINCIPES DE BASE

Afin de pouvoir réaliser les objectifs spécifiés, le système doit être articulé autour des principes de base suivants :

- il faut une documentation centrale et une documentation répartie en usines pour un projet, et localement accessible à l'ensemble des secteurs concernés. Cette documentation sera technique et économique.

- la structuration des données doit être réalisée à deux niveaux:
* le niveau regroupement
* le niveau détail.

A. Le niveau regroupement représente une vision dynamique d'utilisation des données, c'est-à-dire le cheminement intellectuel des personnes réalisant des hypothèses.

B. Le niveau détail représente l'ensemble des informations existantes, réelles résultant des travaux menés par les méthodes à un instant donné.

Entre ces deux niveaux, une consolidation est assurée au travers de relations entre les entités essentielles manipulées.

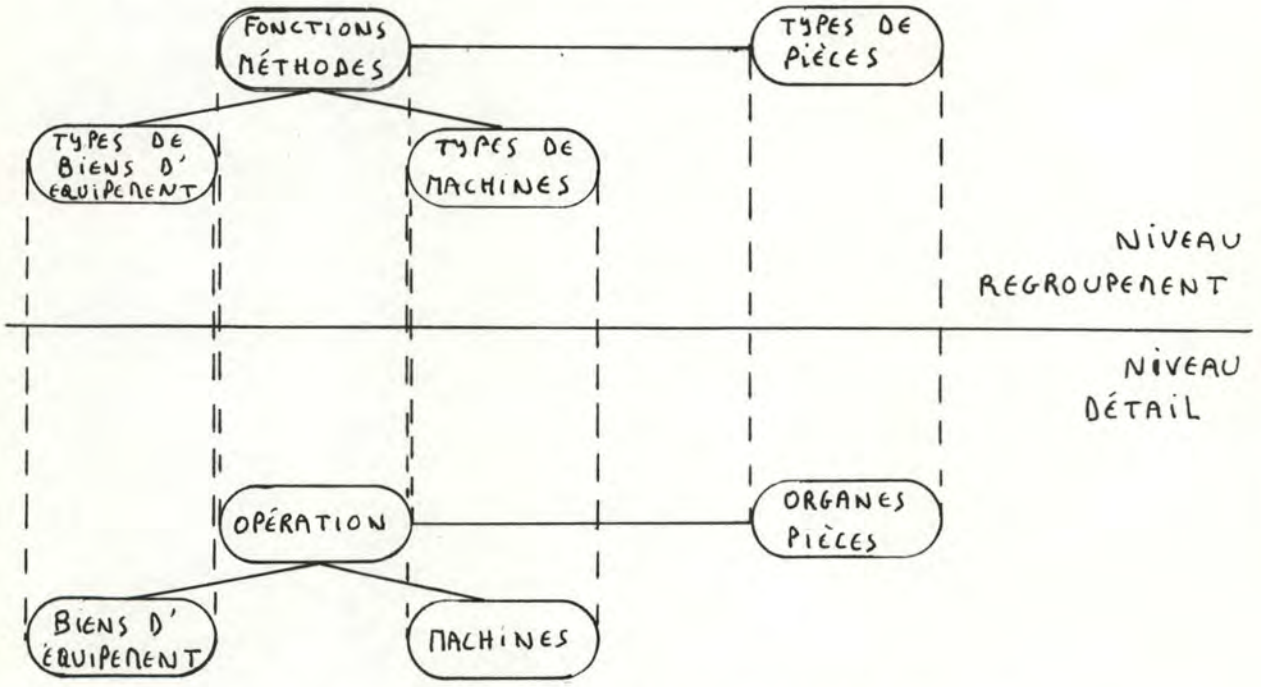


figure 2.1. Les deux niveaux et leur consolidation

2.3. L'APPROCHE RETENUE EN TERME DE SYSTEME D'INFORMATION

Nous présentons les grandes lignes du système à mettre en place.

"L'approche retenue repose sur la volonté de faire du système une image fidèle de l'entreprise en termes d'information et de connaissance. Le système doit représenter au mieux la culture et les activités de l'organisation et servir au mieux les besoins réels des acteurs de l'organisation." [REN-2-85]

Le système utilise une base de données dont les finalités sont :
1) organiser la mémorisation
2) avoir une vision intégrée des données
3) tester des hypothèses.

2.3.1. MEMORISATION

La mémorisation des informations dans la base de données doit permettre non seulement d'améliorer la qualité et la fiabilité de la documentation technique de la filière Méthodes Mécaniques, mais aussi de conserver une partie du **savoir-faire** de la filière.

2.3.2. VISION INTEGREE DES DONNEES

La base de données à construire doit permettre d'avoir une vision intégrée des données.
Les données doivent avoir une sémantique homogène.
Cela permet d'améliorer la communication des informations et des connaissances entre les différentes activités de la filière et éventuellement avec les organisations extérieures.
La vision intégrée des données permet l'amélioration de la communication verticale et de la communication horizontale.

1) Communication horizontale

La base de données doit permettre par exemple que les données soient sémantiquement homogènes entre la Direction de la Planification et des Investissements (D.P.I.) et la Direction des Méthodes Centrales (D.M.C.).
Cette communication permet de formaliser et de faciliter certains circuits informels de données existants.

2) Communication verticale

Signalons tout d'abord que les activités de la Filière Méthodes Mécaniques peuvent se cataloguer dans deux domaines : le domaine des études et le domaine du réalisé.
Le domaine des études correspond à un niveau utilisateur ou privé.
Le domaine du réalisé correspond à un niveau officiel ou public.

La communication verticale consiste à améliorer la continuité entre les deux domaines "étude" et "réalisé".
Ceci permet de tester facilement des hypothèses de travail, ce qui représente un processus extrêmement complexe (ainsi que nous le verrons dans la suite du travail).
Dans le niveau "privé", l'utilisateur formule toutes ses hypothèses, en attendant le choix de l'une d'elles, qui passe alors dans le domaine "public".
Ceci correspond aux deux niveaux de structuration des données détaillés au point 2.2.
Les deux niveaux, regroupement et détail, sont consolidés.
"La consolidation des deux approches a pour objectif d'assurer une continuité entre les deux vues : le domaine "étude" et le domaine "réalisé".
Pour fournir des prévisions de meilleure qualité et plus rapidement, on a besoin d'informations sur le "réalisé" exploitables, c'est-à-dire les informations et leurs liens avec le domaine "étude".
Inversément, la production d'une gamme ne peut se faire sans les hypothèses... élaborées dans la phase prévisionnelle. Ainsi les liens entre les domaines "étude" et "réalisé" doivent être établis". [REN-2-85]

La consultation de la base de données, importante fonction du système, est fortement facilitée et améliorée par la distinction et la consolidation entre les deux niveaux.

2.3.3. ELABORATION DES HYPOTHESES

Nous avons dit que le processus de tests d'hypothèses est complexe.
Néanmoins, on peut aider l'utilisateur à élaborer les hypothèses de travail.

Nous décrivons ci-dessous les aides à l'élaboration des hypothèses.

Il existe des aides passives à l'élaboration d'hypothèses, dont nous donnons la définition, les types, des illustrations, et l'intérêt de les utiliser.

Il existe aussi des aides actives à l'élaboration d'hypothèses. Nous montrons en quoi elles représentent une évolution des aides actives, ainsi qu'un exemple.

A. LES AIDES PASSIVES A L'ELABORATION DES HYPOTHESES

DEFINITION ET TYPES

Les aides passives à l'élaboration des hypothèses de travail sont celles qui interviennent sous la forme de sélections dans la base de données de solutions possibles aux problèmes que se pose l'utilisateur (ou éventuellement de calculs sur des données extraites). Les hypothèses de travail sont soit des hypothèses de correspondance DPI-DMC, soit des hypothèses de nomenclatures, soit des hypothèses de plan de pièces...
Les aides passives à l'élaboration d'hypothèses sont de deux types :
- requêtes passives générales (c'est-à-dire prédéfinies)
- requêtes passives spécifiques (c'est-à-dire introduites par l'utilisateur en fonction de ses besoins présents).

L'aide, qu'elle soit d'un type ou de l'autre, peut se situer à trois niveaux :

niveau 1 : aide à l'élaboration d'hypothèses en faisant intervenir des critères simples de sélection d'informations dans la base de données.
Les critères simples portent uniquement sur le nom de l'entité ou sur des valeurs des attributs de l'entité.

niveau 2 : aide à l'élaboration d'hypothèses en tenant compte de critères liés à l'objet de la sélection dans la base de données. Les critères liés portent sur des associations de l'entité sélectionnée ou sur des valeurs d'attributs des entités reliées à l'entité sélectionnée.

niveau 3 : aide à l'élaboration d'hypothèses en tenant compte de critères croisés, c'est-à-dire manipulant des critères liés et des critères portant sur d'autres données que celles sur lesquelles porte la sélection. Les critères croisés sont en quelque sorte des critères liés aux critères liés à l'objet de la sélection.

REQUETES PASSIVES DE NIVEAU 1

Les requêtes passives de niveau 1 font intervenir, lors de la sélection, des critères simples.

Exemples :

1. sélection de tous les organes de type moteur
2. sélection de tous les véhicules de niveau gamme.
3. sélection de tous les moteurs appartenant à la sous-famille "A1M".

REQUETES PASSIVES DE NIVEAU 2

Les requêtes passives de niveau 2 font intervenir des critères liés à l'objet de la sélection.

Exemples :
1. sélection de tous les moteurs de niveau famille qui reçoivent ou peuvent être montés avec une boîte de vitesses de type "JB".
   Le critère lié est ici le fait que le moteur doit être capable, techniquement, de recevoir la boîte donnée.
2. sélection de toutes les lignes de production qui montent des moteurs.
   Le critère lié est ici le fait que la ligne de production doit être active et monter des organes de type "moteur".

REQUETES PASSIVES DE NIVEAU 3

Les requêtes passives de niveau 3 font intervenir des critères croisés.

Exemple :
sélection des lignes actives qui peuvent encore être engagées dans le montage du moteur A1MU17 pour les besoins de commercialisation de ce moteur en France.
Le critère croisé est le fait que :
1) la ligne de production doit avoir pour chaque période, sa capacité supérieure à son engagement de production actuel.
2) la ligne de production doit être capable de répondre aux besoins de commercialisation de ce moteur.
3) la ligne de production doit être engagée dans le montage d'organes de type moteur.

AUTRES TYPES D'AIDE A L'ELABORATION D'HYPOTHESES

Outre les requêtes passives que nous venons de définir et illustrer, on peut trouver des types d'aide à l'élaboration des hypothèses travaillant par calculs ou contrôles sur des données, plutôt que par sélections de celles-ci.

Exemples :
1. calcul des besoins réels, qui correspondent aux besoins de commercialisation diminués de l'intégration locale.
2. calcul de la cadence, qui correspond au besoin réel divisé par le nombre de jours de la période.

3. contrôle de validité sur les engagements donnés par l'utilisateur.

INTERET DES AIDES PASSIVES A L'ELABORATION DES HYPOTHESES

L'intérêt des aides passives à l'élaboration des hypothèses est évident. Il s'agit d'aide à la décision omniprésente dans l'application, dans par exemple la sélection des solutions possibles de correspondance, la sélection des nomenclatures d'organes possibles, la sélection de pièces possibles composant un organe, la sélection de nomenclatures de pièces possibles, la sélection de lignes pouvant monter ou usiner.

Le mot-clé de ces illustrations est **"possible"**. En effet, l'aide passive à l'élaboration des diverses hypothèses "de travail", sous la forme de sélections dans la base de données, permet non seulement d'offrir à l'utilisateur un choix de solutions **possibles, valides et valables** parmi lesquelles il devra choisir, mais aussi de **réduire considérablement son travail**, puisque les solutions possibles proposées sont le résultat d'un **filtrage efficace et pertinent** de l'extraction des données de la base, en fonction de critères généraux ou spécifiques.

B. LES AIDES ACTIVES A L'ELABORATION DES HYPOTHESES

DEFINITION ET TYPES ; EVOLUTION DES AIDES PASSIVES

Les aides actives à l'élaboration des hypothèses de travail interviennent non plus seulement sous la forme de sélections dans la base de données de solutions possibles aux problèmes que se pose l'utilisateur, mais aussi sous la forme de gestion dynamique et active de la base de données, d'utilisation intelligente de critères déterminés.

Cela se traduit sous la forme de création d'entités et de relations, de propositions à l'utilisateur de solutions **nouvelles**, de filtrage des données sur base de critères variés, multiples et dont le choix d'intervention dans la décision n'est plus fixé, mais variable en fonction de la typologie de problème à résoudre, etc... On retrouve ici notamment la technique de "triggers" employée dans les systèmes de gestion de bases de données actives.

Il est clair qu'il s'agit d'une évolution par rapport aux aides passives développées au point A.

On remarque aisémment que cette évolution, qui n'a pas été implémentée ici, implique, en termes d'outils, le développement de systèmes d'aide à la décision "intelligents", faisant appel à des techniques d'intelligence artificielle.

Les aides actives peuvent être classifiées en deux types de requêtes :

1. les requêtes actives, valorisant les délais, les coûts et les budgets.
2. les requêtes actives, guides dans le choix des hypothèses de nomenclatures.

Nous nous limiterons à donner un exemple de type 2., en montrant l'évolution par rapport à une requête passive.

EXEMPLE DE REQUETE ACTIVE

Lors de l'établissement de nomenclatures possibles d'organes (supposons le raffinement de la famille de moteurs "A" en sous-familles), l'**aide passive** consiste à sélectionner dans la base de données, de façon cachée à l'utilisateur, tous les moteurs de niveau sous-famille, soient A1M, A1N, A2C, A3D, AXZ, AXW, à ensuite en choisir certains qui sont possibles et plausibles, sur base de critères déterminés, pour raffiner la famille de moteurs "A" dans le cadre du montage de celle-ci sur un véhicule dont la boîte de vitesses sera de type JBO et le châssis de type NH3. Soient A1M et A1N qui sont alors proposées à l'utilisateur.

L'**aide active** consiste dans ce cas à suivre la démarche suivante (pour plus de facilité, nous avons personnalisé "le système informatique") :

- à un premier niveau de complexité :

le système, connaissant la signification des codifications, estime que dans le cadre du problème posé, il serait **technologiquement** intéressant de proposer, outre les sous-familles A1M et A1N existant dans la base de données, une **nouvelle** sous-famille A1E, car les caractéristiques codifiées dans le "E" conviennent particulièrement bien à l'adaptation de ce moteur à la boîte de vitesses JBO et au châssis NH3 exigés dans le problème.

- à un second niveau de complexité :

le système ne tient plus seulement compte de l'aspect technologique évoqué ci-dessus, mais aussi d'un aspect **financier**: étant donné les moyens (techniques, humains et financiers) prévus à disposition lors du montage du moteur "A", et étant donné les exigences en termes de nombre de personnes, de durée de fabrication, d'utilisation de machines (nouvelles ou récupérables moyennant adaptations éventuelles), etc..., il est préférable, d'un point de vue financier, de ne proposer que les sous-familles A1M et A1E.

2.4. DESCRIPTION GENERALE DE LA SOLUTION DEVELOPPEE

Nous présentons, en cette fin de première partie, la suite du mémoire.

La solution développée porte uniquement sur l'application "Plans de pièces", laquelle couvre une partie des travaux de coordination réalisés par le sous-système de "Gestion Prévisionnelle" du projet "Filière Méthodes Mécaniques" présenté au premier chapitre (point 1.3.2.).

La deuxième partie du mémoire présente le schéma conceptuel de cette application.
Il s'agit d'une spécification fonctionnelle **détaillée** de l'application "Plans de pièces" du projet "Filière Méthodes Mécaniques".
Cette spécification est établie en application de la méthodologie IDA (Interactive Design Approach) [BOD-83]. Elle est réalisée selon le formalisme du langage DSL (Data Specification Language) [BOD-83].
Le schéma conceptuel est développé en deux parties :
- la première partie concerne le schéma conceptuel des données. Il est décrit au chapitre 3. Le chapitre 4 apporte une justification méthodologique de celui-ci.
- la seconde partie (chapitre 5) décrit le schéma conceptuel des traitements.

La troisième et dernière partie du mémoire porte sur la mise en oeuvre de la solution à l'aide de techniques de prototypage rapide.
Nous utiliserons à cet effet les logiciels ORACLE et DSL-PROTO, que nous comparerons ensuite, dans le cadre du prototypage de l'application "Plans de pièces".

DEUXIEME PARTIE : LE SCHEMA CONCEPTUEL DES DONNEES

INTRODUCTION

La deuxième partie du travail concerne le schéma conceptuel de l'application "Plans de pièces" du projet Filière Méthodes Mécaniques.
Cette partie est composée de trois chapitres.
Un premier chapitre concerne le schéma conceptuel des données.
Sur base du schéma entité-association, on explicite les quatre parties composantes des données de l'application : le descriptif, les besoins de la planification, les nomenclatures et les plans de pièces.
Le deuxième chapitre (chapitre 4) consiste en une justification méthodologique du schéma conceptuel des données. En effet, nous avons rencontré, lors de la construction de ce schéma, deux types de problèmes méthodologiques.
Le premier type de problème concerne la construction du schéma par classification et par groupement, processus que nous définissons et appliquons ensuite à la construction du modèle entité-association de notre application.
Le deuxième type de problème a pour composante la dimension temporelle, à laquelle nous avons du attacher une importance particulière dans le modèle, étant donné l'obligation de prendre en considération la notion de gestion des hypothèses dans l'application "Plans de pièces".

Enfin, le dernier chapitre de cette troisième partie présente le schéma conceptuel des traitements.
Après une description des traitements de l'application "Plans de pièces", nous donnons la structuration et la dynamique des phases, que nous justifions. Puis nous décrivons la phase sur laquelle nous poursuivons l'étude, à savoir la répartition des engagements. Nous en donnons et justifions la statique et la dynamique.
Nous terminons en spécifiant les types d'aide à l'élaboration des hypothèses implémentés dans chaque fonction.

CHAPITRE 3 : LE SCHEMA CONCEPTUEL DES DONNEES

INTRODUCTION

Dans ce chapitre, nous décrivons le schéma conceptuel des données de la partie du sous-système "Gestion Prévisionnelle" (cfr point 1.3.2.) que nous avons choisi de développer dans la suite du travail : l'établissement du "plan de pièces".

Après avoir défini quelques concepts, la partie "Plan de pièces" est décrite dans les grandes lignes. Ensuite, nous approfondissons les notions et les composants du schéma conceptuel des données. Les composants se répartissent en quatre grands types d'information à mémoriser : le Descriptif, les Besoins, les Nomenclatures et les Plans de Pièces.

A cette fin, nous proposons au lecteur le schéma conceptuel des données, support de la description. On trouve en annexe 1 une description plus approfondie (en termes de définitions des entités, attributs, associations, contraintes, connectivités, etc...). La schématisation et la définition des concepts sont basées sur le modèle entité-association largement décrit dans la littérature. On peut à ce sujet consulter [BODART, 83].

Les définitions de concepts importants utilisés dans la partie "plan de pièces" données ci-dessous représentent un **extrait** du schéma conceptuel des données (annexe 1).

3.1. DEFINITIONS

Pour faciliter la lecture de ce chapitre, il nous a semblé utile de rassembler en début de chapitre la définition des principaux concepts utilisés dans la partie "Plan de Pièces". Les définitions sont extraites du schéma conceptuel des données décrit à l'annexe 1.

1) Besoins

Les besoins sont les prévisions sur 7 ans de commercialisation de véhicules, d'organes ou de pièces. Ils sont définis pour un véhicule (ou un organe ou une pièce) et pour un pays (ou une zone de pays). Suivant qu'il s'agit des ventes de véhicules-DPI (ou d'organes-DPI) ou de ventes de véhicules-DMC (ou d'organes-DMC ou de pièces-DMC), on parlera de besoins-DPI ou de besoins-DMC.

2) **Boîte de vitesses**

La boîte de vitesses est un des trois organes de la partie mécanique d'un véhicule automobile.

3) **Cadence**

La cadence est le nombre d'organes montés ou de pièces usinées par jour sur une ligne de production.

4) **Capacité**

La capacité est la quantité maximale de pièces (ou organes) qu'une ligne de production peut usiner (monter).

5) **Châssis**

Le châssis est un des trois organes de la partie mécanique d'un véhicule automobile.

6) **Descriptif**

Le descriptif est l'ensemble des informations concernant les véhicules à construire pendant les sept années à venir; il est établi par le Bureau d'Etudes (B.E.) et la Direction de la Planification et des Investissements (D.P.I.).

7) **Engagement**

L'engagement est la prévision de cadence d'une ligne de production pour un organe ou une pièce.

8) **Hypothèse**

L'hypothèse est un essai de correspondance DPI-DMC, ou de nomenclature, ou de traduction des besoins-DPI en besoins-DMC, ou de plan de pièces, fait par la Direction des Méthodes Centrales (D.M.C.).

9) **Ligne de production**

Une ligne de production est une partie d'une usine qui peut, de façon autonome, usiner une pièce ou monter un organe.

10) **Montage**

Le montage est la fabrication d'un organe sur une ligne de production.

11) **Moteur**

Le moteur est un des trois organes de la partie mécanique d'un véhicule automobile.

12) **Nomenclature**

La nomenclature est un ensemble de liens ou d'hypothèses de liens entre véhicules, organes et pièces, un véhicule étant composé des 3 organes (boîte de vitesses, châssis, moteur) et ceux-ci étant décomposés en pièces, qui peuvent elles-mêmes être composées d'autres pièces.
Elle est établie par la Direction des Méthodes Centrales (D.M.C.). Elle est donc exprimée en termes de véhicules-DMC, organes-DMC (boîte de vitesses-DMC, châssis-DMC, moteur-DMC), pièces-DMC.

13) **Organe**

Un organe est un ensemble mécanique composant un véhicule automobile. Il y a trois organes : boîte de vitesses, châssis et moteur. Un organe est décomposé en pièces.
Différents niveaux de regroupement existent : famille d'organes, sous-famille d'organes, organe proprement dit. On parle d'organe-DPI ou d'organe-DMC, suivant que ce dernier est décrit par la Direction de la Planification et des Investissements (D.P.I.) ou par la Direction des Méthodes Centrales (D.M.C.).

14) **Pays**

Le pays est un lieu de (prévisions de) commercialisation de véhicules, d'organes ou de pièces. Un pays appartient à une zone.

15) **Pièce**

Une pièce est un ensemble mécanique faisant partie d'un organe. Une pièce peut être assemblée (c'est-à-dire composée d'autres pièces) ou nue.
Différents niveaux de regroupement existent : famille de pièces, type de pièces, pièce proprement dite. Une pièce est toujours décrite par la Direction des Méthodes Centrales (D.M.C.), la Direction de la Planification et des Investissements (D.P.I.) ne descendant pas à ce niveau de détail. On ne parle donc que de pièces-DMC.

16) **Plan de pièces**

Le plan de pièces représente les prévisions d'engagements de montage d'organes-DMC et/ou d'usinage de pièces-DMC sur 7 ans, à fabriquer par jour. Il est établi en fonction des besoins-DMC et des capacités des lignes de production.

17) **Plan septennal**

Le plan septennal est un ensemble de 14 périodes correspondant aux 7 années sur lesquelles s'étalent les travaux de la Direction des Méthodes Centrales (D.M.C.) et de la Direction de la Planification et des Investissements (D.P.I.). Une année est divisée en deux périodes de durée à peu près identique.

18) **Regroupement**

Le regroupement, ou raffinage, représente la gradation dans la façon de détailler un véhicule, un organe ou une pièce. On parle de niveaux, de degrés de regroupement.

19) **Usinage**

L'usinage est la fabrication d'une pièce sur une ligne de production.

20) **Véhicule**

Un véhicule est un ensemble composé, du point de vue du constructeur, de trois filières : mécanique, électrique et carrosserie.
Dans la filière mécanique (celle qui nous intéresse), on considère que le véhicule est composé uniquement de trois organes : un moteur, une boîte de vitesses et un châssis (appelé aussi train).
On parle de véhicule-DPI ou de véhicule-DMC, suivant que ce dernier est décrit par la Direction de la Planification et des Investissements (D.P.I.) ou par la Direction des Méthodes Centrales (D.M.C.).
Différents niveaux de regroupement existent pour les véhicules-DMC : gamme de véhicules, famille de véhicules, version de véhicules. Par contre, pour les véhicules-DPI, on parle uniquement de version de véhicules, cette appellation n'ayant aucune corrélation avec celle utilisée pour les véhicules-DMC.

3.2. LA PARTIE "PLANS DE PIECES"

Suivant les critères de structuration définis dans [BODART-83], on peut considérer que la partie "Plans de pièces" est une application. Nous emploierons dès à présent l'appellation **Application Plan de Pièces**.

L'application "Plan de pièces" consiste, dans les grandes lignes, à établir des plans d'engagements de montage d'organes et d'usinage de pièces sur une période de 7 ans, et cela à partir d'un descriptif des véhicules à construire dans les 7 années à venir et d'une prévision de vente de ceux-ci sur 7 ans. Pour établir ces plans d'engagements, appelés plans de pièces, le système informatique traduit le descriptif des véhicules à construire, où ces derniers sont décrits de façon très générale, en nomenclatures plus précises, soit en créant de nouveaux organes et pièces utiles pour les nomenclatures, soit en se servant et en réutilisant des nomenclatures existantes, soit en combinant les deux méthodes de travail. Plusieurs hypothèses de nomenclatures peuvent être établies, auxquelles correspondront autant de plans de pièces.

L'utilisateur peut orienter les hypothèses de nomenclatures, en rejetant les plus improbables, ou se limiter à quelques hypothèses, en fonction de critères de fonctionnalités des véhicules par exemple. Il peut aussi choisir un niveau de raffinement dans l'établissement des hypothèses de nomenclatures : nomenclatures en termes de familles d'organes, d'organes, de familles de pièces, de pièces, etc...

Ensuite, l'utilisateur valide une ou plusieurs hypothèses de nomenclatures, en fonction de deux éléments principalement : la plausibilité des nomenclatures d'une part, l'optimalité des combinaisons d'engagements de production de l'organe ou de la pièce sur différentes lignes de production. Les hypothèses de plans de pièces correspondant aux nomenclatures devenues effectives sont aussi validées.

3.3. DESCRIPTION DU SCHEMA CONCEPTUEL DES DONNEES

3.3.1. INTRODUCTION

Dans l'application "Plans de pièces", quatre grands types d'informations sont à mémoriser :
- les informations contenues dans le **descriptif** des véhicules à construire.
- les informations concernant les **besoins**, les prévisions de ventes de véhicules ou d'organes du descriptif sur 7 ans.
- les informations concernant les **nomenclatures** de véhicules, d'organes ou de pièces.
- les informations contenues dans les **plans de pièces**.

Ces quatre grands types d'informations et les liens, les relations qui les unissent, sont représentés sur le schéma conceptuel (figure 3.2. en fin de chapitre).

La suite de ce chapitre est consacrée à un commentaire analytique du schéma conceptuel des données destiné à mettre en évidence la signification globale qui lui est attachée. Il est nécessaire d'en effectuer la lecture en parallèle avec celle de la représentation graphique du schéma conceptuel (figure 3.2.). Le lecteur consultera chaque fois qu'il le jugera nécessaire les définitions précises des éléments du schéma conceptuel exposées à l'annexe 1.

3.3.2. LE DESCRIPTIF

Le descriptif établi par la Direction de la Planification et des Investissements (D.P.I.) décrit les véhicules et organes à construire dans les 7 années à venir, au point de vue desquels on ne peut encore faire que des supputations en termes très généraux. On parle de véhicules-DPI et organes-DPI.
Le descriptif est composé de codifications ayant une signification bien précise (cfr tableaux des figures 3.3. et 3.4. [fin du chapitre]). Il existe des tables de correspondances entre codification et sémantique de celle-ci (cfr tableau de la figure 3.5. [fin du chapitre]).
Chaque ligne du descriptif correspond aux caractéristiques d'une version de véhicules-DPI. Les versions diffèrent les unes des autres par le modèle de véhicule, la carrosserie, le moteur, la boîte de vitesses, l'énergie (essence ou diesel), l'équipement et le niveau de pollution de l'échappement.
La D.P.I. détermine aussi pour chaque version de véhicules-DPI le moteur, la boîte de vitesses et le châssis qu'elle suppute pour la version de véhicules-DPI.

Les 3 organes-DPI sont dénommés par un code en fonction de leur niveau de regroupement, à savoir famille d'organes, sous-famille d'organes ou organe proprement dit. Des tables de correspondances entre codification et signification existent (voir figure 3.6. en fin de chapitre).
Les autres informations qu'on retrouve sur une ligne du descriptif sont :
- la dénomination de la version de véhicules-DPI ;
- les dates de début et de fin de fabrication ;
- la codification et les observations quant à la présence (obligatoire ou optionnelle) de la direction assistée.

Exemple : prenons la première version de véhicules-DPI du descriptif représenté sur les figures 3.3. et 3.4.
- il s'agit de la version H4 LE2B C
- de moteur A1MU17, de boîte de vitesses NG050H et de châssis 40EAOA.
- cette version est dénommée R18 Berline et est fabriquée jusqu'en mars 83.
- la direction assistée est optionnelle au Canada.

3.3.3. LES BESOINS DE LA PLANIFICATION

Sous ce terme, on regroupe les informations concernant les prévisions sur 7 ans de commercialisations de véhicules définis par la Direction de la Planification et des Investissements (D.P.I.) ou d'organes définis par la DPI.
La DPI établit un plan septennal. Il représente les quantités de véhicules-DPI (par version) et d'organes-DPI à commercialiser (en fonction des études de marché, etc...).
Les quantités à commercialiser sont aussi appelées **besoins**. Le plan septennal est composé de 14 périodes (2 semestres par an).
Par période et par version, les quantités d'organes-DPI sont supérieures ou égales aux quantités de véhicules-DPI car la Régie peut par exemple livrer des organes à d'autres marques de voitures.
Le plan septennal est établi par pays.
On peut considérer qu'il y a deux parties dans le plan septennal:
1. le plan des besoins en véhicules : est établi par version de véhicules-DPI (exemple : figure 3.7. en fin de chapitre : pour le véhicule H4 LE2BXE) et par pays.
2. le plan des besoins en organes : est établi par organe-DPI (châssis, boîte de vitesses, moteur) et par pays.

3.3.4. LES NOMENCLATURES

Les nomenclatures sont un ensemble de liens établis par la Direction des Méthodes Centrales (D.M.C.) entre, dans l'ordre hiérarchique, les véhicules, les organes et les pièces. Elles peuvent soit déjà exister, soit être créées.

A) Les Véhicules

Les véhicules-DMC représentent les véhicules destinés à être commercialisés. Ils sont la traduction des véhicules-DPI (véhicules décrits par la Direction de la Planification et des Investissements) en véhicules dont les caractéristiques sont non plus générales, mais précises, raffinées. L'appellation de ces véhicules correspond de très près à celle connue par le grand public, par les acheteurs (exemple : R 18), et les distinctions entre les véhicules-DMC sont presque toutes basées sur les caractéristiques connues des acheteurs (exemple : berline, break, toit ouvrant, GTL, ...).

Il y a trois regroupements, trois degrés de raffinement dans l'appellation "véhicule" : gamme, famille, version.

* **Gamme de véhicules**

La gamme de véhicules-DMC représente un ensemble de familles de véhicules (nous définirons la famille plus loin) appartenant à la même ligne de produit. Exemple : la gamme R 18.

* **Famille de véhicules**

La famille de véhicules-DMC représente un ensemble de versions de véhicules (nous définirons la version plus loin) ayant les mêmes caractéristiques aux points de vue motorisation et carrosserie. Exemple : la famille R 18 Berline Essence.

* **Version de véhicules**

La version de véhicules-DMC représente un sous-ensemble de la famille de véhicules-DMC. Chaque sous-ensemble (c'est-à-dire chaque version) est déterminée par des caractéristiques plus précises que celles de la famille (exemple : GTL, TL, etc ...) et par la zone géographique de destination du véhicule-DMC (en effet, le climat, l'état des routes et d'autres facteurs peuvent influencer les caractéristiques du véhicule).
Exemple : dans la famille R 18 Berline Essence, on pourra avoir les versions : GTL pour Afrique
TL pour Afrique.

Dans la nomenclature, on retrouve ces véhicules sous forme de codifications (cfr annexe 1).

B) Les Organes

Les 3 organes-DMC (boîte de vitesses-DMC, châssis-DMC, moteur-DMC) ont leurs caractéristiques déterminées par la DMC. Ils ont une utilité pour la DMC dans l'élaboration des plans de pièces.
Les organes-DMC contiennent les mêmes dénominations codifiées que les organes-DPI (cfr annexe 1), ainsi que d'autres informations décrites ci-dessous.

* Une des parties du travail de la DMC, lors de l'établissement de nomenclatures en correspondance avec le descriptif, consiste en la traduction des besoins d'organes-DPI en besoins en organes-DMC (besoins-DMC).
A ce besoin-DMC, il faut ajouter deux coéfficients :

a/ Il est utile de prévoir pour chaque organe-DMC un pourcentage du besoin (c'est-à-dire du nombre d'organes à produire) qu'on ajoutera à ce dernier lors de l'élaboration des plans de pièces, pour couvrir les besoins destinés au Magasin des Pièces de Rechange (MPR).

b/ Lors du montage des organes-DMC sur les lignes de production, il est normal qu'il y ait des erreurs de montage : on a lors du montage des "loupés".
Pour chaque organe-DMC, on peut connaître le pourcentage, le coéfficient d'organes-DMC à ajouter lors de l'élaboration des plans de pièces au besoin pour la commercialisation augmenté du nombre d'organes à prévoir pour le MPR.

* Il y a trois regroupements, trois degrés de raffinement dans l'appellation "organe" : famille d'organes, sous-famille d'organes, organe. Un organe-DMC appartient à une sous famille d'organes, laquelle apppartient à une famille d'organes.
Exemple : le moteur A1MU17 appartient à la sous-famille de moteurs A1M.
La sous-famille de moteurs A1M (et par transitivité le moteur A1MU17) appartient à la famille de moteurs A.

* En fonction des différentes hypothèses de nomenclature, une version de véhicules-DMC peut avoir une ou plusieurs décompositions possibles en chacun des 3 organes-DMC.

C) Les Pièces

Les pièces-DMC ont leurs caractéristiques déterminées par la DMC. Elles ont une utilité pour la DMC dans l'élaboration des plans de pièces.

Les informations concernant les pièces sont :

* Une dénomination (exemple : vilebrequin, pièce 7701023344).

* Comme pour les organes-DMC, on a un coéfficient de "loupés", et un coéfficient "pour le MPR" par pièce-DMC, dont on tiendra compte lors du calcul des besoins en pièces (en fonction de la décomposition des organes en pièces).

* Une pièce-DMC peut avoir différents degrés de raffinement : elle appartient à un type de pièces, lequel appartient à une famille de pièces.
Exemple : la pièce 7701123456 appartient au type de pièces "vilebrequin pour boîte de vitesses JBO". Ce type de pièces (et par transitivité la pièce) appartient à la famille de pièces "vilebrequin".

* En fonction des différentes hypothèses de nomenclatures, un organe-DMC peut avoir plusieurs décompositions possibles en pièces-DMC.
Dans la décomposition d'un organe-DMC en pièces-DMC, il faut faire intervenir, pour un organe et pour une pièce, la quantité de cette pièce faisant partie de cet organe.
Cette quantité aura une influence sur le calcul des besoins en pièces-DMC.
Exemple : la boîte JB0123 est composée de 2 pièces 7700123456 .

* Outre l'appartenance éventuelle d'une pièce-DMC à un organe-DMC, une pièce-DMC peut être composante d'une autre pièce-DMC. On dit qu'une pièce assemblée est composée de pièces nues ou assemblées.
On indiquera pour chaque composition, la quantité d'une pièce composante.
Exemple : la pièce 7700112233 est composée de 3 pièces 7701233244.

* L'appartenance d'une pièce-DMC à une ou plusieurs autres pièces-DMC n'empêche pas son appartenance à un ou plusieurs organes-DMC.

D) Autres considérations

* Etant donné que plusieurs hypothèses de nomenclatures peuvent être établies, on pourra retrouver certaines décompositions identiques de versions de véhicules-DMC en organes-DMC et/ou d'organes-DMC en pièces-DMC et/ou de pièces-DMC en pièces-DMC, tout en ayant des nomenclatures différentes.

Exemple : hypothèse de nomenclature 1 :
- . version de véhicules-DMC : X 40 Berline Essence GTL Afrique.
- . avec un moteur-DMC A1MU17.
- . celui-ci est composé de l'alternateur 7700556677.

hypothèse de nomenclature 2 :
- . pour la même version et le même moteur, l'alternateur est 7700223344.

Lors de chaque décomposition
- d'une version de véhicules-DMC en organes-DMC
- d'organes-DMC en pièces-DMC
- de pièces-DMC en pièces-DMC,

il faut ajouter un numéro d'hypothèse, qui identifiera la décomposition. Ce numéro sera très utile dans l'établissement des plans de pièces, comme nous le verrons plus tard.

* De même, vu que l'application plan de pièces doit se baser sur le descriptif dans l'établissement des nomenclatures (il faut en effet générer différentes hypothèses de nomenclatures plausibles pour décrire en termes d'organes-DMC et de pièces-DMC les véhicules décrits par la DPI), il existe une correspondance entre un organe-DPI et un organe-DMC.
Dans l'établissement des hypothèses de nomenclatures, un organe-DPI pourra correspondre dans la nomenclature à plusieurs organes-DMC, suivant les hypothèses générées.

* Comme nous le verrons au chapitre 4, la nomenclature est un ensemble de liens entre des objets (liens simples, de composition et liens diffus, de regroupement).

3.3.5. LES PLANS DE PIECES

* Nous avons vu qu'à chaque véhicule ou organe décrits par la DPI correspond un "besoin" par pays, c'est-à-dire une prévision de commercialisation sur 7 ans.
Les véhicules-DPI et organes-DPI sont décrits en termes très généraux. Aussi, ils sont traduits, raffinés en véhicules-DMC, organes-DMC et pièces-DMC. Il faut par conséquent traduire les besoins définis par la DPI en besoins définis par la DMC.

* Nous avons vu aussi que les besoins en organes-DMC et pièces-DMC sont calculés par pays, et que différentes hypothèses de correspondance DPI-DMC et de nomenclatures établies par la DMC pouvant exister, différentes hypothèses en besoins-DMC peuvent exister.
La DMC établit différentes hypothèses de plans de pièces, c'est-à-dire différents plans d'engagements de montage d'organes ou d'usinage de pièces, à partir d'une hypothèse de nomenclature DMC à laquelle correspond une hypothèse de besoins-DMC.

* Le plan de pièces peut être établi pour les besoins de l'organe ou pièce en question correspondant à un pays (ou une zone de pays, ou l'ensemble des pays).
Un plan de pièces est donc établi
- pour un organe (une pièce)
- pour un pays
- pour une hypothèse de nomenclature (d'où pour une hypothèse de besoins).

* Pour le montage d'un organe sur une ligne de production (l'usinage d'une pièce), on détermine 14 engagements (1 par période du plan septennal). Chaque engagement représente une prévision de la quantité de cet organe à monter (pièce à usiner) par jour sur cette ligne de production.

* L'appellation "plan de pièces" est trop restreinte. Il s'agit en fait des plans d'organes-DMC et/ou de pièces-DMC. Ils peuvent être établis à n'importe quel niveau de raffinement.

Exemple : on pourra avoir le plan pour la famille de boîtes de vitesses JB, le plan pour le type de pièces "vilebrequin pour boîte JB", le plan pour le moteur A1MU17, le plan pour la pièce 7700224466, dans l'hypothèse H1 et pour le pays Belgique (figure 3.1.).

Organe : moteur A1MU17
Pays : Belgique
Hypothèse : numéro 1

	P1	P2	P3	....	P14
Besoins	1000	2000	600		2000
Engagements					
Le Mans :	400	0	100		400
Fasa :	600	2000	500		1600

figure 3.1. : exemple de plan de pièces

* Il est possible de connaître, pour un pays et pour une pièce-DMC (un organe-DMC) la quantité (par période du plan septennal) de cette pièce-DMC (organe-DMC) que ce pays peut usiner (monter).
En effet, toutes les pièces-DMC (organes-DMC) ne sont pas usinées (montés) par la Régie. Certains pays usinent (montent) des pièces (organes).
On considère cette quantité, qu'on appelle intégration locale, constante tout au long du plan septennal. Cette intégration locale pourra être soustraite des besoins lors du calcul des engagements.

* Une ligne de production est désignée par son nom. Exemple : Le Mans 1. Elle contient deux autres informations :
. un échéancier à terme : c'est un ensemble de 14 capacités théoriques (1 par période du plan septennal). On suppose qu'on connait pour chaque ligne de production la capacité théorique de chacune des périodes. Elles sont prévues, fixées par la DMC.
. la capacité à terme : quantité maximale de pièces (organes) qu'une ligne peut usiner (monter).
Les capacités ne varient pas en fonction de l'organe ou de la pièce.

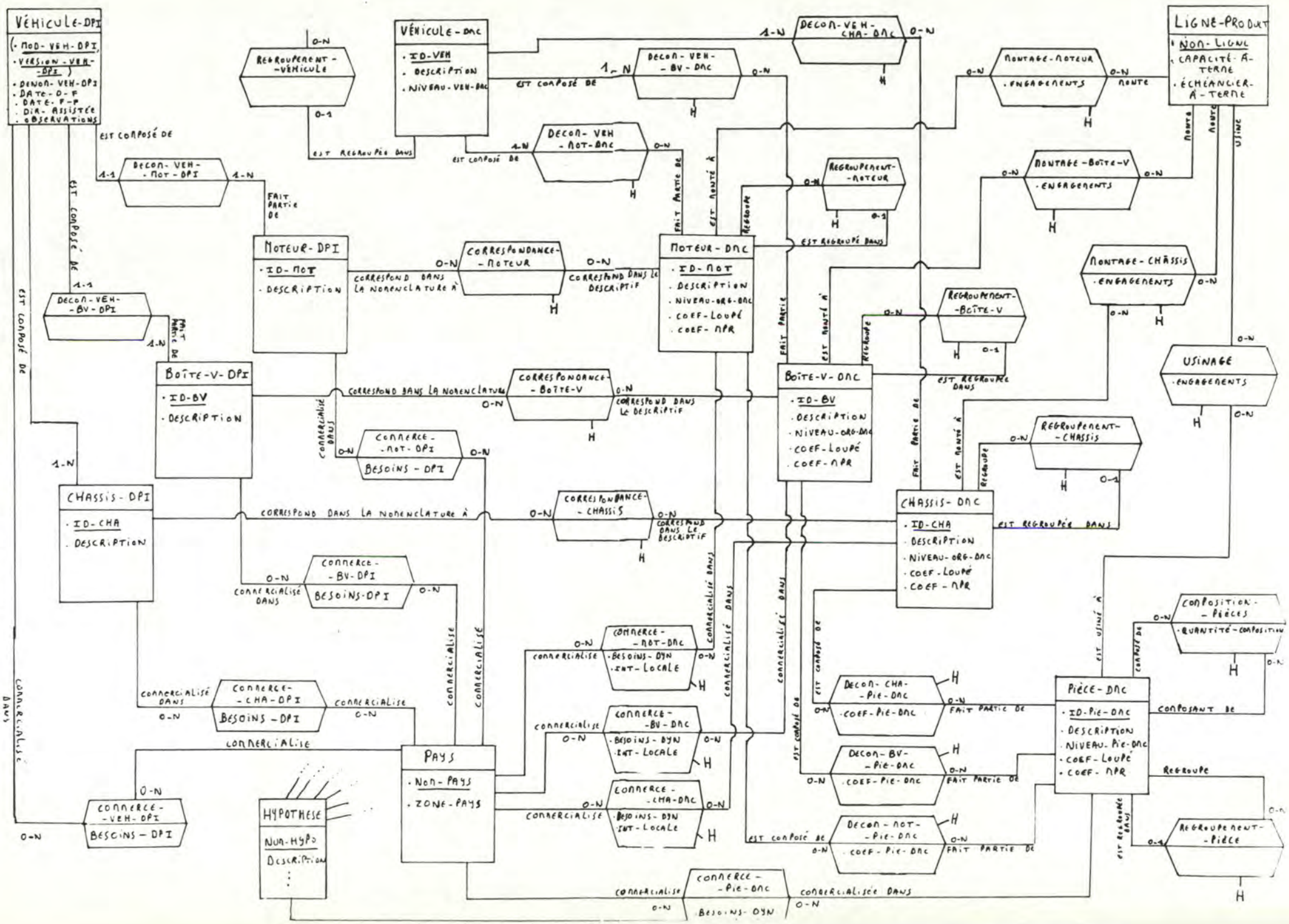


figure 3.2. Le schéma conceptuel des données

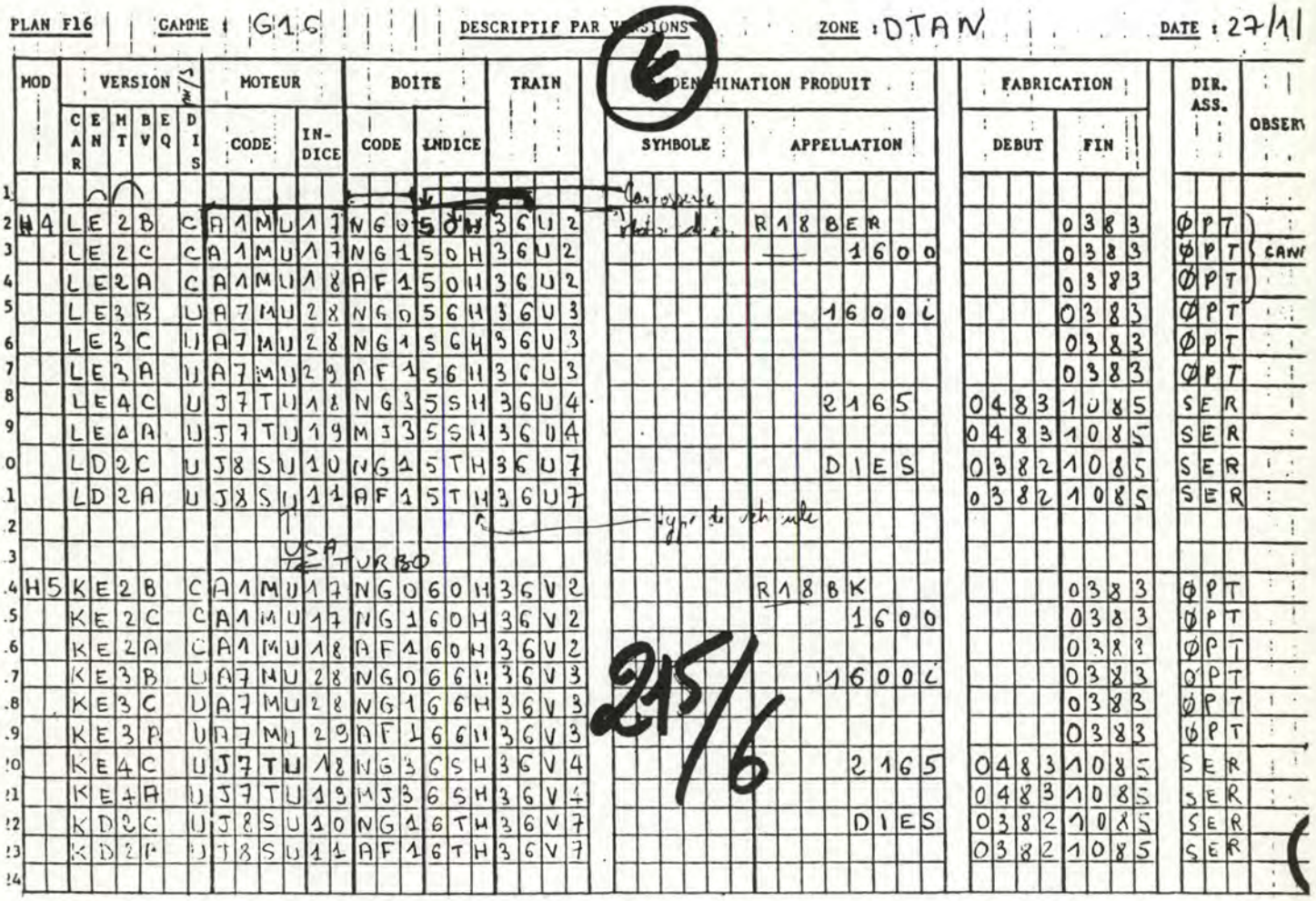


figure 3.3. Descriptif (état brut)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
MODÈLE		VERSION						MOTEUR						BOÎTE						CHÂSSIS					
								CODE				INDICE		CODE			INDICE								
		CAR	EN	MT	BV	EQ	DIS	FAMILLE	SOUS	FAMILLE	OPTION	CAR	BV	FAMILLE	SOUS FAMILLE		CAR	NOT	GAMME VEH						
H	4	L	E	2	B		C	A	1	Π	U	1	7	N	G	O	5	O	H	4	O	E	A	O	A
H	4	L	E	4	C		U	J	7	T	U	1	8	N	G	3	5	5	H	4	A	Π	G	H	E

27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	...	79
DÉNOMINATION												FABRICATION								DIRECTION ASSISTÉE			OBSERVATIONS		
												DÉBUT				FIN									
R	1	8		B	E	R	L	i	N	E						0	3	8	3	O	P	T	CA	NA	DA
R	1	8		B	E	R		2	1	6	5	0	4	8	3	1	0	8	5	S	E	R			

figure 3.4. Descriptif (clarifié-codé)

2 - Identification des boîtes de vitesses

Définies par leur Nom et leur Indice suivant le principe ci-dessous :

DEFINITION BV			
NOM = 3 Caractères			INDICE
Lettre	Lettre	Chiffre (*)	3 Chiffres
.Désigne la famille de mécanisme. .Une même famille regroupe les boîtes qui possèdent le plus grand nombre de pièces en commun. .Les familles sont constituées comme suit : BVA : ACDEFGM BVM : petit entr'axe ligne d'arbres: HJKL BVX : entr'axe moyen NPQRST BVM : gros entr'axe UVWX Boîtes diverses : BYZ	.Caractèrise le pont lié à l'architecture de la boîte. A : 354 B : 384-391... CD : Libres E : 139 F : 139 C G : 330-336... H : Peugeot J : 376 K : Libre L : 389 M : Libre N : 367/369 O : Sans pont P : 141 Q : 141 RVI R : 397 S : 329 T....Z : Libres	.Définit les variantes par type de boîte donné. *Un chiffre pair correspond à un nombre pair de vitesses et inversement : 0,2,4... 4vitesses 1,3,5... 3 ou 5 vitesses 6,7,8... véh. 4X4 Ce chiffre croit avec le couple dans un type de boîte pont donné.	Comme précédemment, il identifie complètement la boîte avec toutes ses particularités.

Exemples :

BVM	384	4 vitesses	:	JB0
BVM	391	5 vitesses	:	JB1
BVM	352	4 vitesses	:	NG0
BVM	385	5 vitesses	:	NG1
BVM	383	4 vitesses	:	NG2 (gros couple)
BVM	393	5 vitesses	:	NG3 (gros couple)

figure 3.5. Table de correspondance codification-sémantique

Direction des Méthodes Centrales
Mécaniques et Forges
Méthodes Boites de vitesses - Sce 0078

RENAULT
Automobiles
J.C. GLEMAREC

PLAN F17 - CODIFICATION B.V

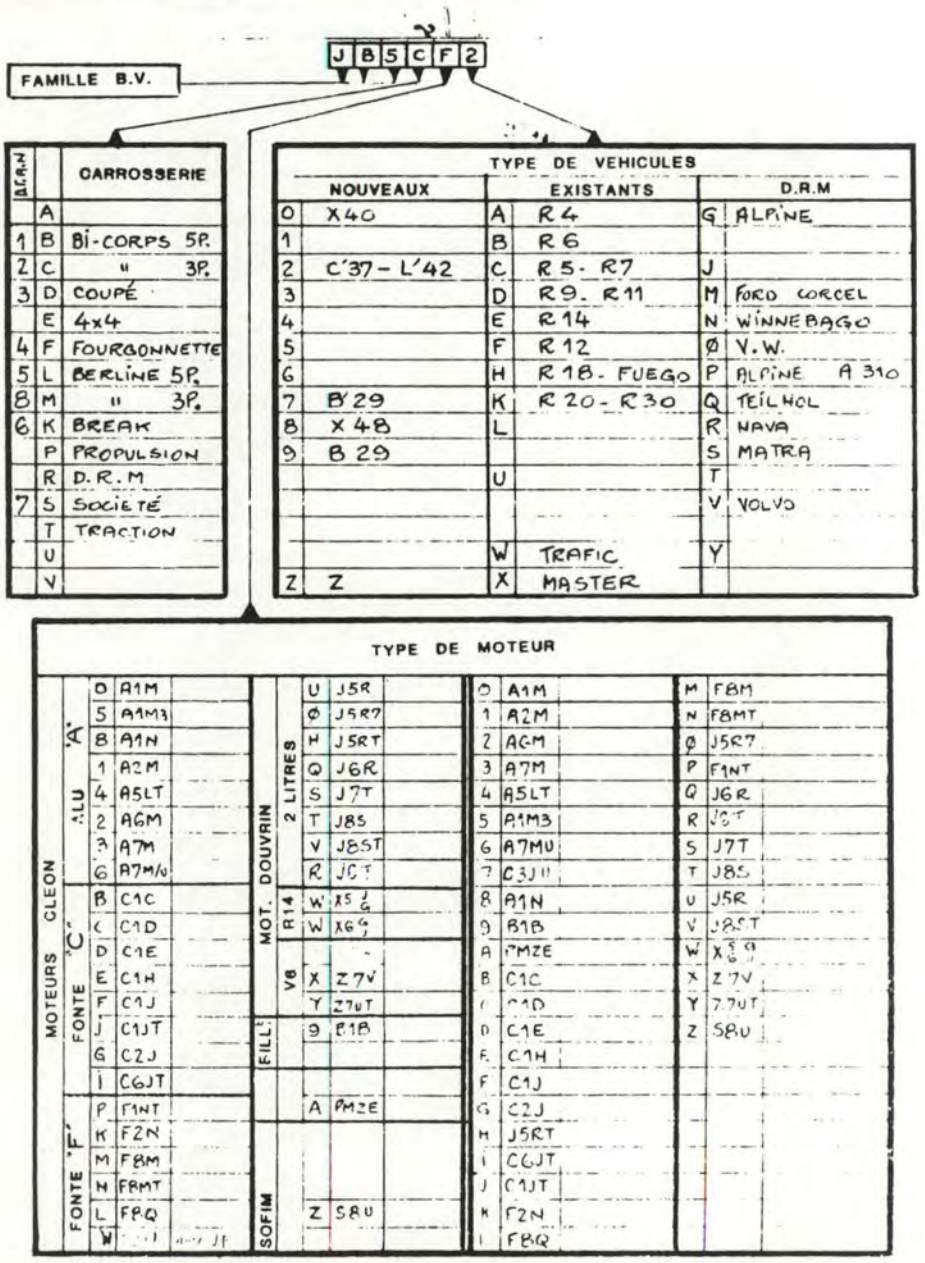


figure 3.6. Table de correspondance codification-signification

PLAN SEPTENNAL DES
BESOINS DU VÉHICULE - DPI : H4 LE2BXE

ANNÉES / PAYS	1986		1987		1988		1989		1990		1991		1992	
	1 [1]	2 [2]	1 [3]	2 [4]	1 [5]	2 [6]	1 [7]	2 [8]	1 [9]	2 [10]	1 [11]	2 [12]	1 [13]	2 [14]
FRANCE	1.000	2.000	3.000	1.200	1.300	1.500	1.000	900	0	0	0	0	0	0
BELGIQUE	1.000	1.000	2.000	2.000	3.000	3.000	3.500	3.500	1.000	1.000	800	800	700	700
LUXEMBOURG	125	125	120	120	110	110	95	95	75	75	50	50	20	20

figure 3.7. Plan septennal des besoins

CHAPITRE 4 : JUSTIFICATION METHODOLOGIQUE DU SCHEMA CONCEPTUEL DES DONNEES

INTRODUCTION

Lors de la construction du schéma conceptuel des données de l'application "Plan de Pièces", diverses techniques de modélisation ont été utilisées.

Ces techniques peuvent être regroupées sous deux aspects principalement.

Le premier aspect concerne la méthodologie de construction.

Nous parlons dans ce chapitre de la construction d'un schéma conceptuel des données par classification et hiérarchisation, et la construction par groupement hiérarchique, principes utilisés dans notre exemple.

A cette fin, nous reparlons d'abord de la nomenclature (décrite précédemment) en tant qu'ensemble de liens. Ces liens, structurés, peuvent influencer la technique de construction du schéma conceptuel des données.

Le deuxième aspect concerne la dimension temporelle dans la construction du schéma conceptuel des données de l'application "Plans de Pièces".

L'optique de base de données temporelle a du être choisie dans ce cadre pour répondre à la notion de gestion des hypothèses dans l'application "Plans de Pièces", ce dont nous parlons dans ce chapitre.

4.1. CONSTRUCTION PAR CLASSIFICATION ET PAR GROUPEMENT

Dans ce point, nous abordons la méthodologie de construction. Avant de définir précisémment la classification et le groupement et de montrer les principes de construction suivant ces méthodes, nous montrons que la nomenclature établie par les Méthodes Mécaniques est un ensemble de liens, et nous montrons comment les agents des Méthodes considèrent ces liens.

4.1.1. LA NOMENCLATURE, UN ENSEMBLE DE LIENS

RAPPEL : La Direction des Méthodes Centrales (DMC) est amenée à traduire le descriptif en termes de véhicules, d'organes et de pièces définis de manière plus précise.
On parlera de . véhicule-DMC,
. moteur-DMC, boîte de vitesses-DMC, châssis-DMC (ce sont les 3 organes-DMC)
. pièce-DMC.
Ces objets constituent la nomenclature des Méthodes.

Explicitons ce terme de **NOMENCLATURE**.

Ce terme recouvre un ensemble de liens entre des entités.
Suivant l'interlocuteur des Méthodes Centrales (Coordination, Montage, Usinage), ce terme couvre le même concept mais ne couvre pas le même domaine.

* Le coordinateur devra connaître la correspondance entre véhicules-DPI/organes-DPI et véhicules-DMC/organes-DMC. A partir d'une nomenclature générale portant sur une famille /gamme / version de véhicules, le coordinateur devra disposer des hypothèses de constitution plus précises de véhicules futurs en famille d'organes, sous-famille d'organes, organes proprement dits.

* Le monteur et l'usineur seront amenés à avoir une décomposition plus fine, à avoir des pièces intermédiaires. Les monteurs travailleront sur les organes, familles de pièces, types de pièces et pièces ; les usineurs plus souvent sur les types de pièces et pièces (assemblées et nues).

Nous avons dit : nomenclature = ensemble de liens.
On note deux types de liens. La terminologie est celles utilisée par les agents des Méthodes.

*** liens simples ou de composition :**

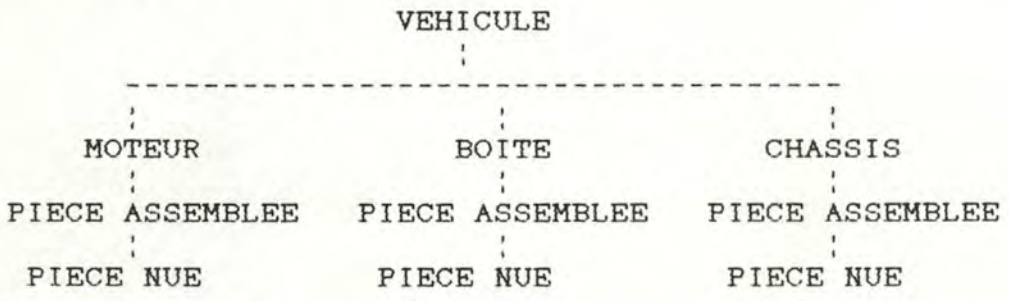


figure 4.1. liens simples ou de composition

* **liens diffus ou de regroupement, de raffinage :**

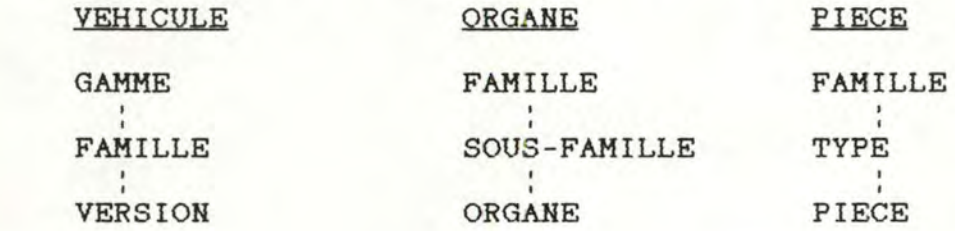
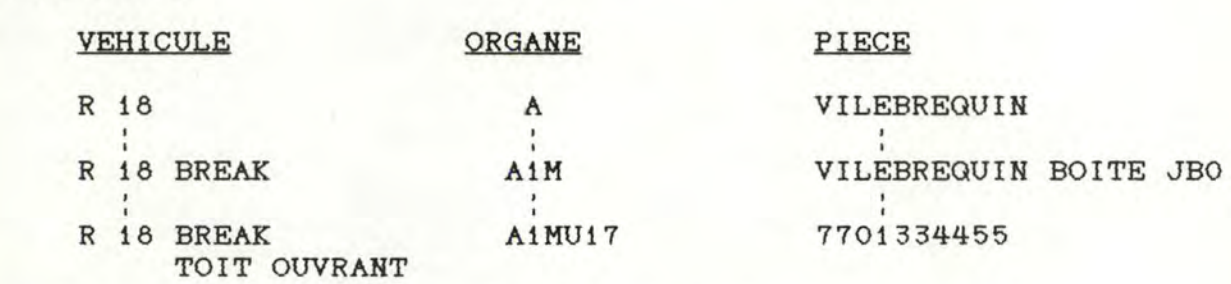


figure 4.2. liens diffus ou de regroupement

Exemple :



N.B. : organe = boîte, châssis, moteur.

figure 4.3. exemple de liens diffus, de regroupement

Dans ce cas, on a un ensemble de liens entre des entités à différents niveaux de définition.

Ces liens (particulièrement les liens diffus) sont la matérialisation de la PROGRESSIVITE entre le "PREVISIONNEL" et le "REALISE", entre le domaine des "ETUDES" et le domaine du "REALISE" . Ils représentent les choix des concepteurs du véhicule.

Ainsi, l'"organe" MOTEUR passera durant le temps de sa conception
du niveau "famille de moteurs"
au niveau "sous-famille de moteurs"
puis au niveau "moteur proprement dit".

Cette représentation de l'évolution temporelle est étudiée plus profondément au point 4.2.

La figure 4.4. représente les différents liens.

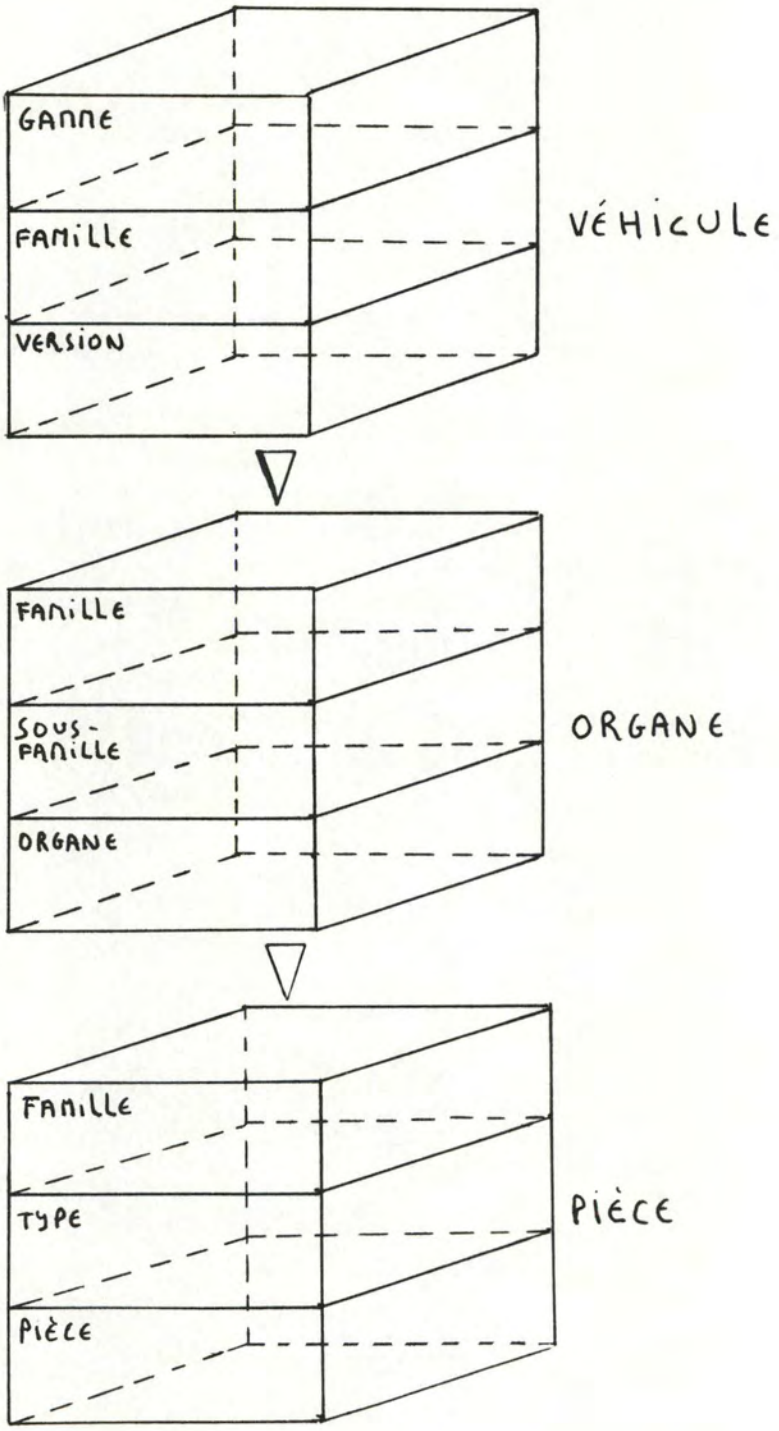


figure 4.4. vue tridimensionnelle de la nomenclature

4.1.2. DEFINITIONS : CLASSIFICATION ET GROUPEMENT

Donnons les définitions des processus de classification et de groupement.
Ces définitions sont inspirées de [FUR-86] et [BRO-82].

A. CLASSIFICATION

La classification est une forme d'abstraction dans laquelle un ensemble d'objets est considéré comme une classe d'objets de niveau supérieur.
La relation employée s'appelle "est-partie-de".

B. GROUPEMENT

Le groupement est une forme d'abstraction dans laquelle une relation entre des objets membres est considérée comme un objet de niveau supérieur.
La relation employée s'appelle "est-membre-de".

Par rapport aux deux types de liens que nous avons expliqué au point 4.1.1., on peut établir les deux correspondances suivantes, que nous développons respectivement aux points 4.1.3. et 4.1.4. :
* les liens simples traduisent un processus de classification
* les liens diffus traduisent un processus de groupement.

4.1.3. CONSTRUCTION PAR CLASSIFICATION ET HIERARCHISATION

Soient les concepts suivants à modéliser :
- un véhicule, ensemble mécanique composé de trois organes : boîte de vitesses, châssis et moteur.
- une boîte de vitesses, un châssis et un moteur (appelés organes), ensembles mécaniques faisant partie d'un véhicule et composés de pièces assemblées.
- une pièce assemblée, ensemble mécanique faisant partie d'un organe et composée de pièces nues.
- une pièce nue, élément mécanique faisant partie d'une pièce asemblée.

Entre ces concepts, existent des liens simples, de composition.
Le schéma, dans ce cas, se construit selon un double processus d'abstraction : la définition de classes et l'établissement d'une hiérarchie sur ces classes. Chaque classe regroupe des objets de la nomenclature possédant les mêmes caractéristiques. On obtient la mise en évidence des types d'entités véhicule, moteur,

châssis, boîte de vitesses, pièce. Ces types d'entités sont reliés par des associations qui traduisent les liens hiérarchiques ou liens simples, de composition.
On obtient une hiérarchie de classifications, chaque classification représentant le fait qu'une entité d'un certain niveau d'abstraction **fait partie** d'une entité d'un niveau d'abstraction supérieur.

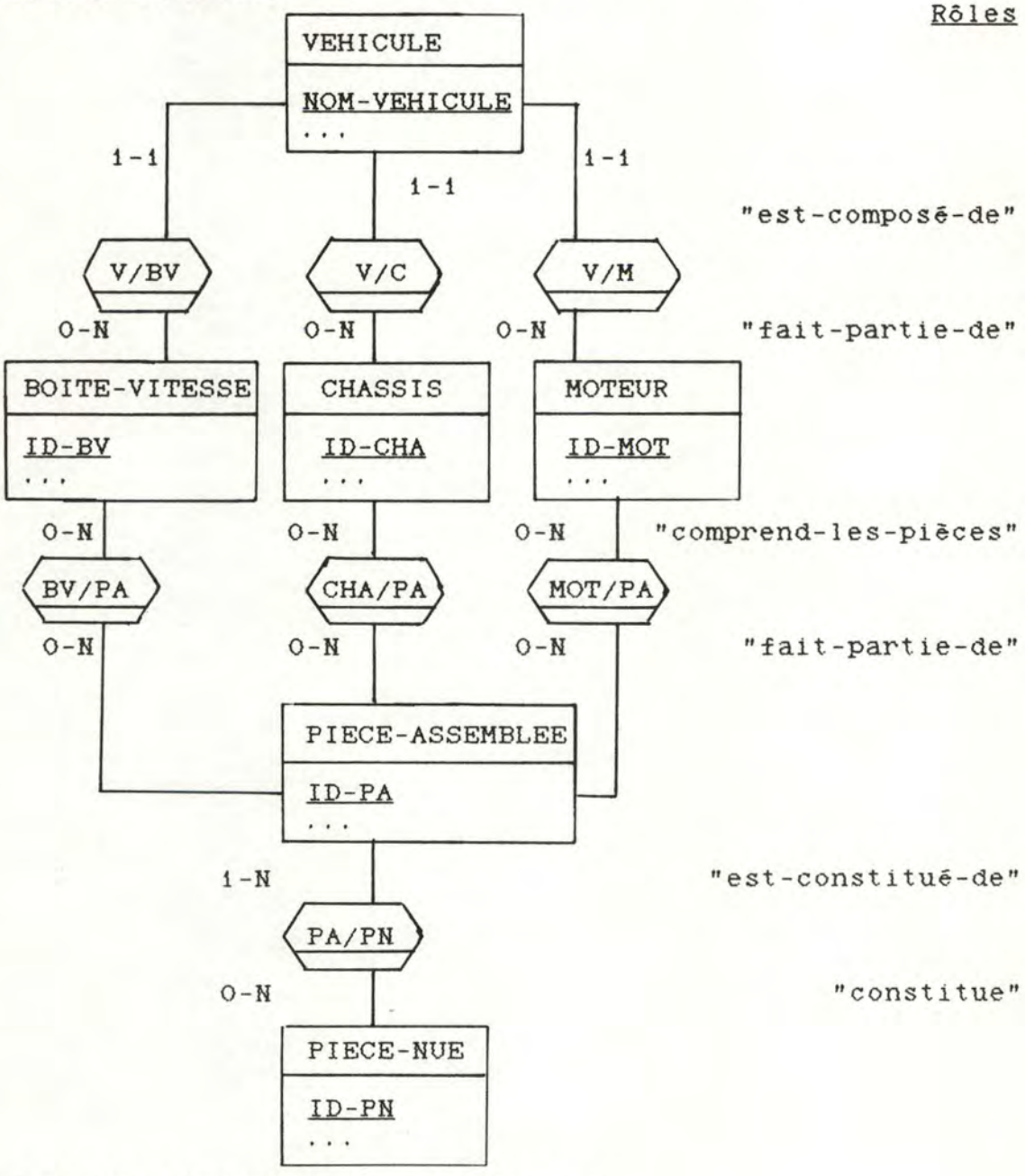


figure 4.5. Hiérarchie de classifications

Remarque : - la distinction entre pièces assemblées et pièces nues n'étant pas toujours reconnue, nous avons implémenté celle-ci sous forme d'une relation de composition portant sur une seule entité pièce (cfr figure 3.2.).
- les connectivités diffèrent par rapport à celles présentées au chapitre 3, étant donné que nous n'avons pas pris en considération ici la gestion des hypothèses. On considère donc par exemple qu'un véhicule est composé d'une seule boîte de vitesses, d'un seul châssis et d'un seul moteur.

INTERET DE LA CONSTRUCTION PAR CLASSIFICATION ET HIERARCHISATION

L'intérêt de cette démarche de construction réside dans le fait qu'elle peut être appliquée de manière systématique et rigoureuse lorsque les informations, les données faisant l'objet de la conceptualisation existent à différents niveaux d'abstraction, lorsque les données appartiennent à un domaine évolutif. On tient compte de deux paramètres : le temps et/ou le niveau hiérarchique, ce dernier d'un point de vue macroscopique.
Les données évoluent vers un niveau de plus en plus détaillé avec le temps.
C'est le cas dans l'application "Plan de pièces", dans les concepts que nous avons à modéliser.
Chaque niveau d'abstraction représente la vue par un utilisateur des données du système d'information, étant donné ses besoins, étant donné le niveau de détail auquel il doit descendre.
Dans notre cas, certains agents des Méthodes travaillent sur le véhicule, sans s'intéresser aux parties qui le composent, certains agents utilisent le véhicule et ses trois parties, etc...

4.1.4. CONSTRUCTION PAR GROUPEMENT HIERARCHIQUE

Soit le concept suivant à modéliser :
un véhicule, ensemble mécanique existant à différents niveaux de raffinage, de regroupement : gamme de véhicule, famille de véhicule, version de véhicule.

On peut définir et modéliser de la même manière un organe et une pièce.

Entre ces concepts, existent des liens diffus, ou de regroupement, de raffinage.
Le schéma, dans ce cas, se construit par le processus de groupement, chaque groupement représentant le fait qu'une entité **est membre** d'une entité d'un niveau d'abstraction supérieur.

En effet, gamme de véhicule, famille de véhicule et version de véhicule sont membres de l'entité véhicule.

La figure 4.6. présente la modélisation du groupement.

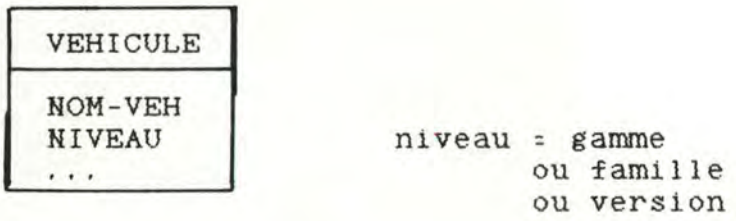


figure 4.6. Modélisation du groupement

L'attribut "niveau" représente le critère de regroupement.
On peut introduire une relation de regroupement, qui permet de modéliser un **concept supplémentaire**, à savoir qu'une gamme de véhicule est composée de familles de véhicule, et qu'une famille de véhicule est composée de versions de véhicule.
On introduit donc une **hiérarchie de groupements**. Cette hiérarchie est représentée à la figure 4.7.

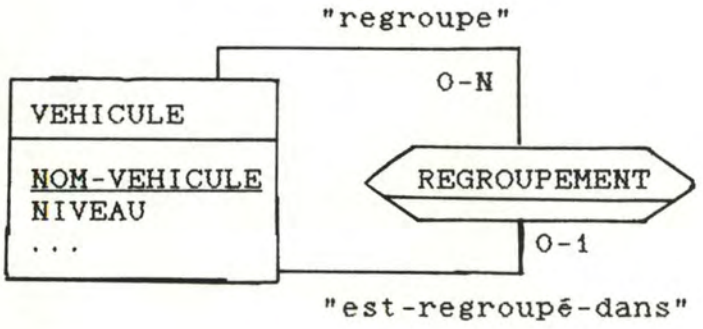


figure 4.7. Hiérarchie de groupement

De nouveau, on peut dire que cette modélisation permet d'offrir à l'utilisateur une vision des données adaptée à ses besoins, puisque certains utilisateurs travaillent au niveau de la gamme, d'autres au niveau de la famille, etc...

4.2. LA DIMENSION TEMPORELLE DANS LA CONSTRUCTION DU SCHEMA

4.2.1. GESTION DES HYPOTHESES DANS L'APPLICATION PLAN DE PIECES

A. DESCRIPTION

La description de la gestion des hypothèses dans l'application "Plans de Pièces" constitue en partie un rappel de ce qui a déjà été évoqué au point 3.3. Ce rappel est simple et incomplet, mais néanmoins suffisant pour traiter le problème envisagé ici.
Signalons d'abord que lorsqu'on parle d'utilisateur, il s'agit d'une personne employée à la Direction des Méthodes Centrales (DMC) et chargée de l'établissement de la nomenclature des Méthodes et de l'élaboration des plans de pièces.

A partir du descriptif des véhicules et des organes associés prévus pour les 7 années à venir, établi par la Direction de la Planification et des Investissements (DPI), la Direction des Méthodes Centrales (DMC) établit des hypothèses de correspondance entre les organes décrits par la DPI et des organes qui seront réellement fabriqués (organes-DMC, c'est-à-dire moteur-dmc, boîte de vitesses-dmc et châssis-dmc). Elle établit aussi des hypothèses de nomenclature, de décomposition de ces organes-DMC en pièces-DMC, des hypothèses de décompositions de pièces-DMC en d'autres pièces-DMC et des hypothèses de regroupements entre différents niveaux de définition des organes-DMC (famille, sous-famille, organe proprement dit) et des pièces-DMC (famille, type, pièce proprement dite).

Plus tard, les utilisateurs des Méthodes Centrales pourront demander d'établir le plan de répartition des engagements de telle pièce ou de tel organe sur différentes lignes de production (usinage ou montage). C'est ce qu'on appelle le plan de pièces. Pour une même pièce (un même organe), le plan de pièces pourra être demandé dans différentes hypothèses de nomenclatures, c'est-à-dire dans différentes hypothèses d'appartenance de la pièce (de l'organe) à un ensemble constituant un organe, un véhicule,...
Toutes les hypothèses de nomenclatures, et tous les plans de pièces (en fait toutes les hypothèses de plans de pièces) sont établis par des gens des Méthodes de façon interne, dans leur "espace de travail", sans aucune diffusion "officielle". Les utilisateurs appellent cela le **domaine privé**, qui n'est accessible que d'eux.

Le travail de l'utilisateur consiste ensuite à valider une hypothèse de nomenclature en fonction de la cohérence, de la performance et de l'optimalité de celle-ci, et pour chaque organe/pièce de cette nomenclature, à valider un plan de pièces en fonction de la faisabilité des engagements de fabrication.

Telle nomenclature devient alors effective, valide, et on supprime les autres hypothèses émises. Les engagements correspondants deviennent aussi valides, effectifs, et on supprime les autres hypothèses d'engagements.
Les hypothèses validées passent du domaine privé au **domaine public**, c'est-à-dire qu'elles deviennent effectives, valides, qu'elles sortent de l'espace de travail de la D.M.C. et donc qu'elles sont accessibles à tous.

La notion de **domaine privé** et **domaine public** est importante pour les utilisateurs.

B. ROLE DU NUMERO D'HYPOTHESE

Le numéro d'hypothèse est un concept introduit dans un but de traitement de l'application. En effet, dans le système d'information décrit, ce concept n'existe pas en tant que tel.
Les utilisateurs de la D.M.C. ne connaissent pas cette notion de numéro d'hypothèse. Ils travaillent plutôt sur la notion de **domaine privé** et **domaine public**.

Le numéro d'hypothèse permet de modéliser l'**évolution** des réflexions des utilisateurs des Méthodes, la **progressivité** de la construction des nomenclatures et des plans de pièces.

Le concept d'hypothèse ne constitue pas pour l'organisation un concept autonome dans le cadre du référentiel considéré.
La notion d'hypothèse sous-entend un traitement, elle représente un espace de travail de l'utilisateur.

C'est un concept non reconnu, non identifié au sein de la D.M.C. On peut parler de concept fugitif dans le temps.

4.2.2. CORRELATION AVEC LES BASES DE DONNEES TEMPORELLES

A. DEFINITION D'UNE BASE DE DONNEES TEMPORELLE

Une base de données temporelle est une base de données qui prend en compte les aspects de temps, de gestion de celui-ci, par opposition à une base de données statique qui décrit une réalité à l'instant courant.
Une base de données historique (ou temporelle) est donc une succession de bases de données statiques.

Quant aux primitives de manipulation des données dans une base de données temporelle, remarquons qu'il n'y a pas de mise à jour des données, car on cumule des renseignements sur des données, que l'ajout d'un fait dans la base revient à compléter ou à créer un objet et que la suppression d'un objet n'existe pas physiquement, mais est traduite par l'enregistrement de cessation d'existence de l'objet dans le cadre du référentiel pris en compte par la base de données temporelle.

Il existe différents mécanismes permettant de traduire les aspects temporels sur lesquels nous ne nous étendrons pas.
Signalons néanmoins que le temps est une entité à part entière.

B. TEMPS ET NUMERO D'HYPOTHESE

Si l'on se réfère à la définition d'une base de données temporelle, on constate que la prise en compte de la gestion du numéro d'hypothèse est un concept "artificiel" introduit pour gérer les aspects de temps. En effet, on désire avoir une trace de l'évolution des nomenclatures et des plans de pièces.
On ne désire pas mettre à jour ces nomenclatures, ces plans de pièces, mais cumuler ceux-ci, créer un historique de ceux-ci, afin de pouvoir choisir plus tard l'hypothèse la plus plausible.
Le schéma conceptuel des données de l'application "Plans de Pièces" correspond donc à une modélisation prenant en compte l'aspect de gestion du temps, cet aspect prenant dans ce cas la forme de gestion d'hypothèses.

4.2.3. CONCLUSION

Nous avons introduit la notion de temps (sous forme de notion d'hypothèse) que les utilisateurs de la D.M.C. ne gèrent pas de façon formelle. Cela nous a obligé à prendre l'optique d'une base de données temporelle.

Le **temps** est représenté par une entité HYPOTHESE. Celle-ci est identifiée par un attribut numéro d'hypothèse.
Cet attribut numéro d'hypothèse représente la **date**.

Chaque relation de correspondance DPI-DMC, de décomposition, de montage, etc... est une association TERNAIRE sur d'une part les deux entités sur laquelle la relation porterait si la base de données était statique, d'autre part sur l'entité hypothèse.

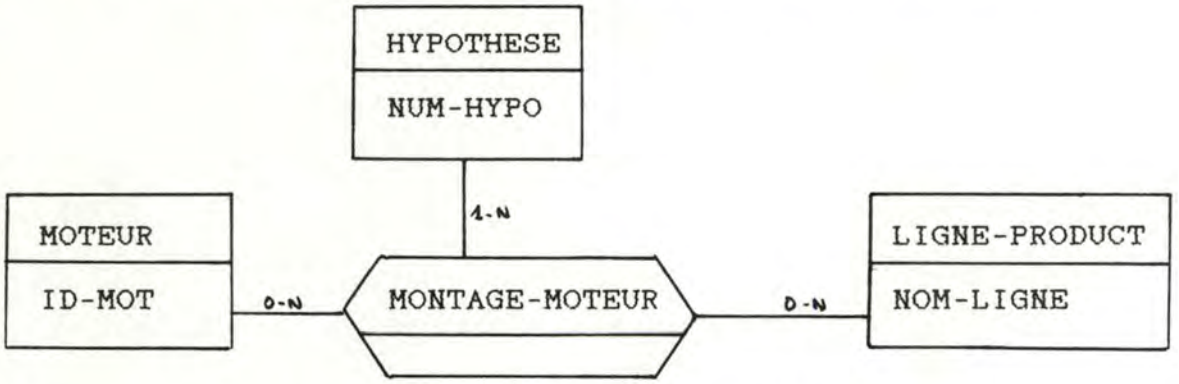


figure 4.8. modélisation de la gestion des hypothèses

CHAPITRE 5 : LE SCHEMA CONCEPTUEL DES TRAITEMENTS

INTRODUCTION

L'objectif de ce chapitre est de décrire le schéma conceptuel des traitements. A cette fin, il contient d'abord une description des traitements dans l'application "Plans de pièces". On trouve ensuite l'énumération des phases de l'application "Plans de pièces" et la dynamique de celles-ci.

Le chapitre contient ensuite une description de la phase "Répartition-des-engagements" (phase 16), qui a été choisie pour être découpée en fonctions (il aurait été trop long de répéter ce processus pour chaque phase), ainsi que la dynamique et la statique des fonctions de la phase de répartition des engagements.

5.1. TRAITEMENTS DANS L'APPLICATION PLANS DE PIECES

REMARQUES PRELIMINAIRES

1. La description des traitements ci-dessous ne peut être comprise que si le lecteur a mémorisé les concepts, définitions et principes clés de l'application "Plans de pièces", décrits au chapitre 3.

2. Néanmoins, il nous a semblé utile de remettre l'accent sur deux concepts :
   - lorsqu'on parle d'utilisateur, il s'agit d'une personne employée à la Direction des Méthodes Centrales (D.M.C.) et chargée de l'établissement de la nomenclature des Méthodes et de l'élaboration des plans de pièces.
   - "l'hypothèse courante" signifie : une incrémentation du numéro d'hypothèse initial, ayant été utilisé lors de l'établissement de la corespondance entre des organes-DMC et les organes-DPI du véhicule-DPI choisi en début d'exécution.

3. Dans la description des traitements de l'application "Plans de pièces" ci-dessous, beaucoup d'hypothèses simplificatrices ont été faites. Elles ne modifient pas la réalité des traitements, mais font abstraction de considérations, de problèmes, de détails dont l'explication et la description auraient été d'une part trop longues, d'autre part inutiles dans le cadre de la résolution des problèmes soulevés ici.

La Direction des Méthodes Centrales (D.M.C.) reçoit de la Direction de la Planification et des Investissements (D.P.I.) un descriptif des véhicules à construire dans les 7 années à venir et d'une prévision des ventes de ceux-ci sur 7 ans. La D.M.C. choisit dans le descriptif un véhicule-DPI sur lequel travailler, ainsi que les organes-DPI y attachés.

Ensuite, la D.M.C. établit une correspondance entre les organes-DPI et des organes-DMC. Il n'y a pas d'ordre d'exécution requis pour l'établissement des trois correspondances : boîte de vitesses, châssis, moteur. Pour la boîte de vitesses et le châssis, plusieurs solutions possibles de correspondance sont proposées. Parmi ces solutions, l'utilisateur en choisit une ou plusieurs, faisant ainsi différentes hypothèses de correspondance ; si aucune des solutions possibles proposées ne lui convient, il peut créer un ou plusieurs organes-DMC afin d'établir les hypothèses de correspondance. Il peut combiner les deux techniques. L'utilisateur mettra à jour les besoins (= les prévisions de vente) des organes-DMC mis en correspondance, en se basant sur les besoins du descriptif associés aux organes-DPI. Pour le moteur, il n'y a qu'une solution possible : le moteur-DPI est mis en correspondance avec un moteur-DMC de même niveau de regroupement et de même dénomination. L'utilisateur n'a donc pas le choix. La mise-à-jour est effectuée de la même manière que pour boîte de vitesses et châssis.

Lorsque les 3 correspondances DPI-DMC ont été établies, le système sélectionne tous les organes-DMC de niveau famille ayant été mis en correspondance avec un organe-DPI du véhicule-DPI choisi. (il y en a 3 au maximum : un moteur-DMC, une boîte de vitesses-DMC, un châssis-DMC). Parmi les familles d'organes-DMC, l'utilisateur choisit celles qu'il veut raffiner, c'est-à-dire celles pour lesquelles il désire faire des hypothèses de décompositions en sous-familles d'organes-DMC.

Ensuite, pour chaque famille choisie, le système sélectionne toutes les sous-familles qui pourraient la raffiner et propose ainsi à l'utilisateur des hypothèses de nomenclatures, de décompositions de la famille d'organes en sous-familles d'organes. L'utilisateur choisit une ou plusieurs solutions possibles proposées et/ou établit de nouvelles hypothèses de nomenclature, en créant de nouveaux organes de niveau sous-famille.
En fonction des besoins de vente de la famille d'organes-DMC, il y a mise-à-jour des besoins de vente des sous-familles liées par l'hypothèse courante.

Après l'élaboration des nomenclatures de décomposition en sous-familles pour chaque famille d'organes-DMC choisie, le système sélectionne tous les organes-DMC de niveau sous-famille,

soit ayant été mis en correspondance avec un organe-DPI du véhicule-DPI choisi (il y en a trois au maximum), soit faisant partie d'une hypothèse courante de nomenclature (cfr ci-dessus).
Parmi les sous-familles d'organes-DMC, l'utilisateur choisit celles qu'il veut raffiner en organes-DMC proprement dits.

Ensuite, pour chaque sous-famille choisie, le système sélectionne tous les organes-DMC proprement dits qui pourraient la raffiner et propose ainsi à l'utilisateur des hypothèses de nomenclatures, de décompositions de la sous-famille d'organes-DMC en organes-DMC proprement dits.
L'utilisateur choisit une ou plusieurs solutions possibles proposées et/ou établit de nouvelles hypothèses de nomenclatures, en créant de nouveaux organes-DMC proprement dits.
En fonction des besoins de vente de la sous-famille d'organes-DMC, il y a mise-à-jour des besoins de vente de l'organe-DMC proprement dit.

La même successsion d'actions (choix par l'utilisateur de solutions - proposées et/ou créées -, ensuite élaboration d'hypothèses de liens - nomenclatures -) peut être appliquée ensuite et successivement à :
- la décomposition d'organes-DMC en pièces-DMC (quel que soit le niveau de regroupement des organes et des pièces, pourvu que les contraintes décrites en annexe 1 soient respectées).
- le regroupement de pièces-DMC (nomenclatures en termes de familles de pièces, types de pièces et pièces proprement dites).
- la composition de pièces-DMC en d'autres pièces-DMC (quel que soit le niveau de regroupement des pièces-DMC, pourvu que les contraintes décrites en annexe 1 soient respectées).

Lorsque l'élaboration de ces hypothèses de nomenclature est terminée, le système sélectionne tous les triplets (organe/pièce, pays/ensemble de pays, numéro d'hypothèse) possibles pour lesquels l'utilisateur peut établir un plan de pièces; parmi ces possibilités, l'utilisateur en choisit une ou n'en choisit pas (s'il ne désire pas faire de plan de pièces).

Si l'utilisateur choisit d'établir un plan de pièces (c'est-à-dire une hypothèse de répartition des besoins en engagements sur différentes lignes de production), le système sélectionne les lignes de production qui peuvent monter l'organe-DMC choisi ou usiner la pièce-DMC choisie (cette sélection se faisant en fonction de différents critères et contraintes décrits en annexe 1), ce qui permet à l'utilisateur d'une part de faire les hypothèses de répartition des engagements sur les lignes de production proposées (ou créées par lui si nécessaire), d'autre part de valider ou non l'hypothèse établie si la répartition donne ou non satisfaction.

Si l'utilisateur décide de valider l'hypothèse de plan de pièces, celui-ci devient effectif (c'est-à-dire que les engagements sur les lignes de production deviennent effectifs, existants) et les hypothèses de nomenclatures qui ont amené à ce plan de pièces deviennent des nomenclatures effectives, existantes.
Sinon, l'utilisateur peut rechoisir un plan de pièces à effectuer parmi les triplets proposés à nouveau.

5.2. DECOUPE EN PHASES

5.2.1. STRUCTURATION

L'application "Plans de pièces" peut être découpée en 17 phases, ci-dessous énumérées. Ces phases sont ensuite décrites, ainsi que les critères d'identification. Puis nous justifions les critères d'identification.

1. prise-en-compte-du-véhicule-DPI
2. création-correspondance-boîte-de-vitesses-DPI-DMC
3. création-correspondance-châssis-DPI-DMC
4. création-correspondance-moteur-DPI-DMC
5. sélection-des-familles-organes-de-la-correspondance
6. élaboration-nomenclatures-de-sous-familles-organes
7. sélection-des-sous-familles-organes
8. élaboration-nomenclatures-organes
9. sélection-des-organes-décomposables
10. élaboration-des-décompositions-organes-pièces
11. sélection-des-pièces-raffinables
12. élaboration-des-nomenclatures-de-pièces
13. sélection-des-pièces-composables
14. élaboration-des-compositions-de-pièces
15. sélection-des-plans-de-pièces-possibles
16. répartition-des-engagements
17. mise-à-jour-des-hypothèses

Pour chaque phase identifiée, on spécifie s'il s'agit d'un traitement manuel, automatisable ou interactif, puis on décrit brièvement le traitement. Entre deux phases différentes, on indique le(s) critère(s) d'identification de rupture de phases.

1. prise-en-compte-du-véhicule-DPI

Traitement interactif : lorsqu'on reçoit l'identification des véhicules-DPI, on recherche les organes-DPI attachés au véhicule-DPI choisi par l'utilisateur.

Critère : disponibilité de toutes les informations au moment du déclenchement.

2. création-correspondance-boîte-de-vitesses-DPI-DMC

Traitement interactif : choix de certaines solutions possibles proposées et/ou choix de nouvelles solutions ; création de la correspondance DPi-DMC et mise-à-jour des besoins.

Critère : disponibilité de toutes les informations au moment du déclenchement.

3. création-correspondance-châssis-DPI-DMC

Traitement interactif : idem que phase 2, mais pour les châssis.

Critère : disponibilité de toutes les informations au moment du déclenchement.

4. création-correspondance-moteur-DPI-DMC

Traitement interactif : établissement de la correspondance entre le moteur-DPI et un moteur-DMC, avec création éventuelle de ce dernier, et mise-à-jour des besoins.

Critère : point d'attente (accumulation)

5. sélection-des-familles-organes-de-la-correspondance

Traitement interactif : sélection de tous les organes de niveau famille ayant été mis en correspondance avec l'organe-DPI du véhicule-DPI choisi ; parmi ces familles, choix par l'utilisateur de celles qu'il veut raffiner.

Critère : point de décision

6. élaboration-nomenclatures-de-sous-familles-organes

Traitement interactif : pour l'organe de niveau famille, on sélectionne toutes les sous-familles qui pourraient le raffiner ; choix par l'utilisateur de certaines nomenclatures proposées et/ou choix de nouvelles solutions, création de la nomenclature et mise-à-jour des besoins.

Critère : point d'accumulation

7. sélection-des-sous-familles-organes

Traitement interactif : sélection de tous les organes de niveau sous-famille, soit ayant été mis en correspondance avec un organe -DPI du véhicule-DPI choisi, soit faisant partie d'une hypothèse courante de nomenclature ; parmi ces sous-familles, choix par l'utilisateur de celles qu'il veut raffiner.

Critère : point de décision

8. élaboration-nomenclatures-organes

Traitement interactif : pour l'organe de niveau sous-famille, sélection de tous les organes qui pourraient le raffiner ; choix par l'utilisateur de certaines nomenclatures proposées et/ou choix de nouvelles solutions, création de la nomenclature et mise-à-jour des besoins.

Critère : point d'accumulation

9. sélection-des-organes-décomposables

Traitement interactif : sélection de tous les organes de n'importe quel niveau de regroupement, qui ont été liés par l'hypothèse courante de correspondance DPI-DMC ; choix par l'utilisateur, parmi les organes proposés, de ceux qu'il veut décomposer en pièces.

Critère : point de décision

10. élaboration-des-décompositions-organes-pièces

Traitement interactif : sélection des pièces pouvant faire partie de la décomposition de l'organe choisi ; choix par l'utilisateur de certaines décompositions possibles et/ou choix de nouvelles décompositions, création de la nomenclature et mise-à-jour des besoins.

Critère : point d'accumulation

11. sélection-des-pièces-raffinables

Traitement interactif : sélection de toutes les pièces de niveau famille ou type, qui ont été liées à des organes par l'hypothèse courante de correspondance DPI-DMC ; choix par l'utilisateur des pièces proposées qu'il veut raffiner.

Critère : point de décision

12. élaboration-des-nomenclatures-de-pièces

Traitement interactif : sélection de tous les types de pièces et pièces pouvant faire partie de la nomenclature ; choix par l'utilisateur de nomenclatures possibles proposées et/ou choix de nouvelles nomenclatures, création de la nomenclature et mise-à-jour des besoins.

Critère : point d'accumulation

13. sélection-des-pièces-composables

Traitement interactif : sélection de toutes les pièces pouvant être composées d'autres pièces, ayant été liées à des organes par l'hypothèse courante de correspondance DPI-DMC ; choix par l'utilisateur des pièces proposées qu'il veut composer.

Critère : point de décision

14. élaboration-des-compositions-de-pièces

Traitement interactif : sélection de toutes les pièces pouvant faire partie de la composition de la pièce choisie ; choix par l'utilisateur de compositions possibles proposées et/ou choix de nouvelles compositions, création de la composition et mise-à-jour des besoins.

Critère : point d'accumulation ; point de décision

15. sélection-des-plans-de-pièces-possibles

Traitement interactif : sélection de tous les triplets possibles (organe, pays, numéro d'hypothèse) permettant d'élaborer un plan de pièces ; choix par l'utilisateur d'un plan de pièces à élaborer ou choix de ne plus élaborer de plan de pièces.

Critère : point de décision

16. répartition-des-engagements

Traitement interactif : sélection des lignes pouvant monter/ usiner; choix par l'utilisateur de lignes possibles proposées et/ ou création de nouvelles lignes, répartition des engagements sur ces lignes ; choix par l'utilisateur de valider le plan de pièces.

Critère : point de décision

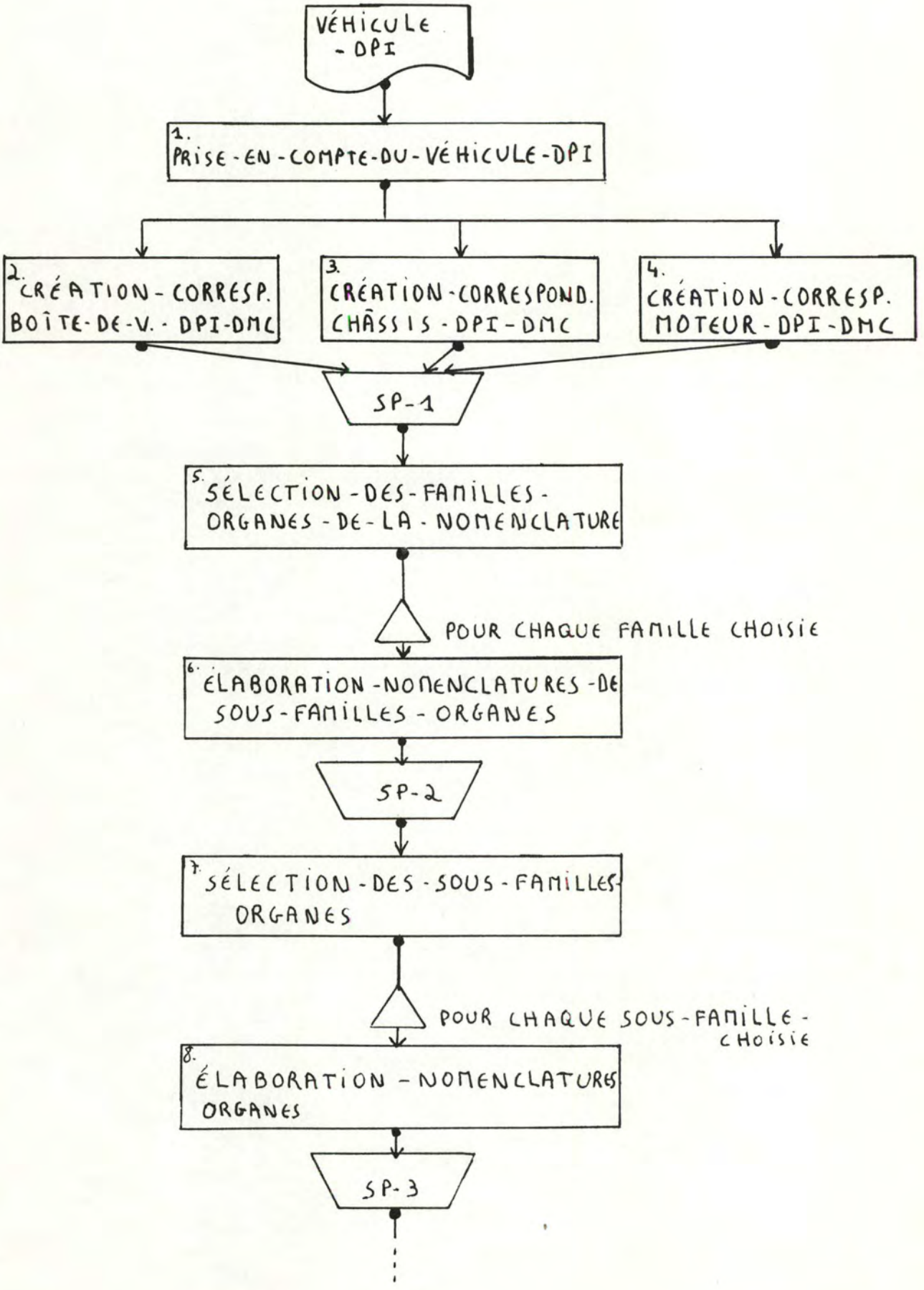
17. mise-à-jour-des-hypothèses

Traitement automatisable : mise-à-jour des hypothèses, qui deviennent des nomenclatures existantes et des productions effectives.

CRITERES D'IDENTIFICATION

Le critère principal d'identification des phases est l'unité temporelle. Il y a en général, dans notre application, rupture de phases parce qu'il y a un point d'attente tel qu'un point de décision humaine ou un point d'accumulation.
Néanmoins, on peut prendre en considération un autre critère d'identification : l'absence de changement d'objet sur lequel porte le traitement, puisque celui-ci diffère parfois en fonction de l'objet et/ou du niveau de regroupement de l'objet.
Il y a par conséquent rupture de phases dès qu'on change d'organe (exemple : trois phases différentes pour la création de la correspondance) ou dès qu'on change de niveau de regroupement d'un organe (exemple : élaboration des nomenclatures de sous-familles d'organes et élaboration des nomenclatures d'organes sont deux phases différentes).
On constate d'ailleurs une répétitivité dans l'enchaînement des phases : celles-ci sont regroupées en blocs portant sur des entités différentes et des types de traitement différents. On a par exemple un bloc formé des phases 5 et 6, qui permet la décomposition de familles d'organes en sous-familles d'organes. Puis on a un bloc (phases 7 et 8) effectuant le même traitement à un niveau de regroupement inférieur (décomposition des sous-familles d'organes en organes proprement dits).

5.2.2. DYNAMIQUE



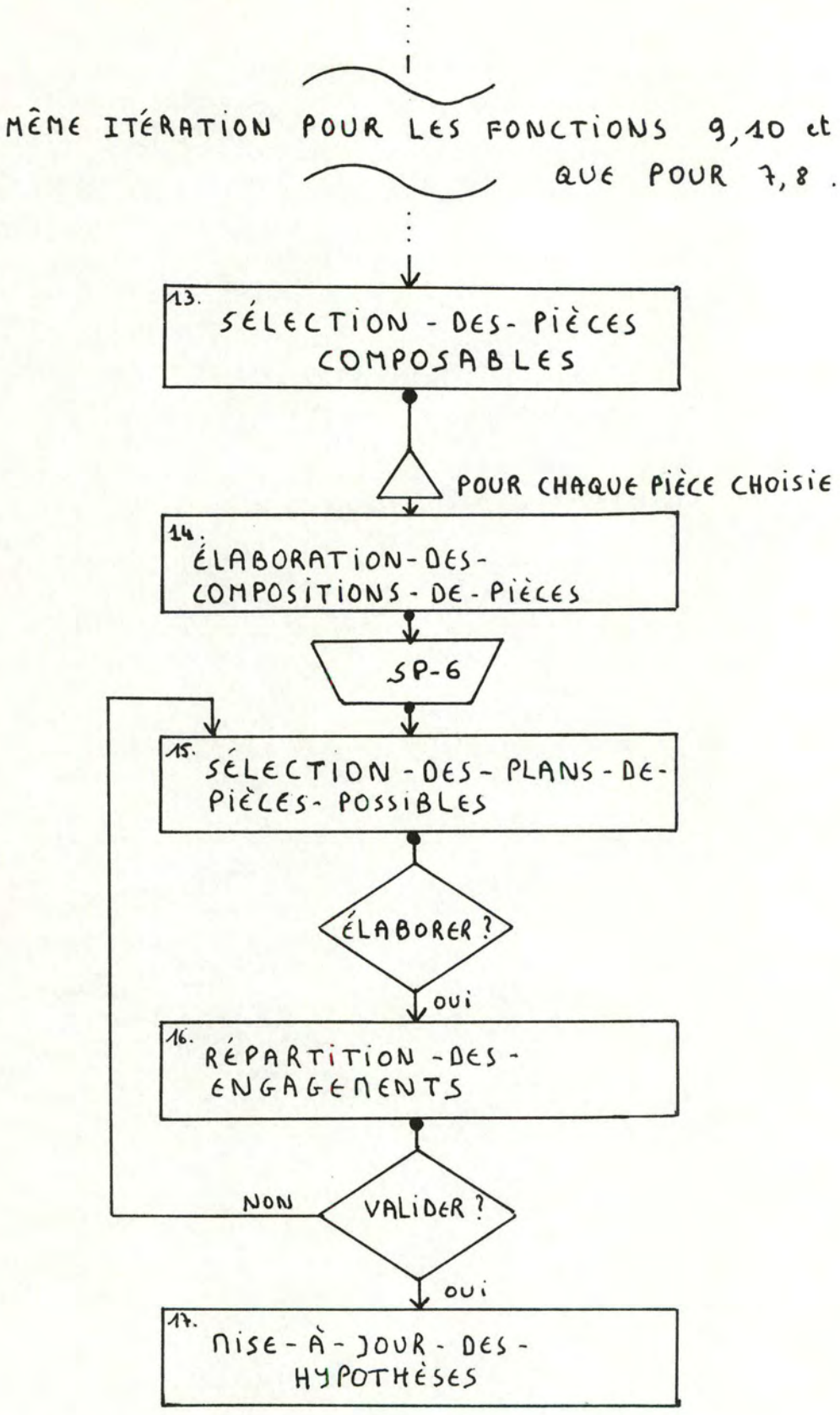


figure 5.1. Dynamique des phases

On peut justifier la dynamique de la façon suivante :
ainsi que nous l'avons dit au point 5.2.1., on travaille par bloc portant sur un traitement à effectuer sur une ou plusieurs entités.
Du point de vue dynamique, les blocs représentent la séquentialité suivante :

```
sélection d'occurrences d'un objet (1)
pour chaque occurrence de l'objet (1) choisie
     traitement de cette occurrence
fin du traitement de toutes les occurrences de l'objet (1)
```

Il est clair que nous avons pris l'optique, dans la solution informatique, d'avoir la dynamique des traitements continue, séquentielle.
Chaque bloc de traitements (défini ci-dessus) suit le bloc précédent.
Dans la réalité, l'utilisateur peut très bien débuter au niveau de la phase 9 par exemple, à savoir la sélection des organes décomposables, sans avoir besoin d'être passé par les phases antérieures, que ce soit en terme de dynamique ou de statique des traitements.

5.3. DESCRIPTION DE LA PHASE "REPARTITION DES ENGAGEMENTS"

Le processus de répartition des engagements lors de l'élaboration des plans de pièces se déroule de la façon décrite ci-dessous.

Dans les lignes de production existantes, deux types peuvent être choisis pour servir à la répartition des engagements des besoins d'un organe ou d'une pièce pour un pays (ou un ensemble de pays) (et pour un numéro d'hypothèse) (on ne tiendra pas compte de ces deux paramètres ici).

1. Les lignes de production actives, mais pouvant encore être engagées, c'est-à-dire dont pour chaque période, l'engagement est inférieur à la capacité et dont le type d'objet déjà produit (boîte de vitesses, châssis, moteur, pièce) est le même que celui sur lequel on élabore un plan de pièces actuellement.
2. Les lignes de production inactives, c'est-à-dire qui ne sont pas engagées dans le montage d'organes ou l'usinage de pièces.

Le système sélectionne donc ces lignes de production.

Ensuite, le système recherche les 14 besoins (un par période du plan septennal) correspondant à l'organe ou à la pièce et au pays pour lesquels le plan de pièces est à faire. Il traduit ces besoins en cadences (c'est-à-dire qu'il divise les besoins d'une période par le nombre de jours de la période).

L'utilisateur choisit alors parmi les lignes actives et/ou inactives proposées, celles sur lesquelles il compte engager tout ou partie des besoins. Il y a aussi vérification que les lignes données par l'utilisateur existent bien.

Le système doit aussi permettre à l'utilisateur de créer de nouvelles lignes de production, soit parce que l'ensemble des lignes existantes n'est plus capable de répondre aux exigences d'engagements des besoins, soit parce que l'utilisateur désire mettre en activité des lignes nouvelles.

L'utilisateur doit ensuite répartir les engagements. A cette fin, pour chaque ligne de production choisie ou créée, le système doit permettre à l'utilisateur de donner un engagement pour chaque période (tout en visualisant les cadences requises).
Si la ligne de production est active, il s'agit d'une mise-à-jour des engagements. Sinon, il s'agit d'une création des engagements.

Remarque : on ne tient pas compte ici de la soustraction d'une éventuelle intégration locale.

Le système contrôle ensuite la validité des engagements d'une ligne de production introduits par l'utilisateur, par rapport aux besoins et aux capacités. Lors du contrôle, les contraintes suivantes doivent être respectées :
pour chaque période :

- engagement inférieur ou égal à capacité (engagement existant + nouvel engagement s'il s'agit d'une ligne de production active).

- engagement inférieur ou égal à besoin.

Si une ou plusieurs erreurs sont détectées, l'utilisateur doit les corriger, c'est-à-dire mettre à jour les engagements erronés.

Lorsque la répartition des engagements a été effectuée pour toutes les lignes de production choisies ou créées, le système contrôle la validité des engagements introduits par l'utilisateur pour l'ensemble des lignes de production choisies ou créées, par rapport aux besoins, à savoir qu'il faut respecter la contrainte:

somme des engagements pour toutes les lignes inférieure ou égale à besoin (pour chaque période).

S'il n'y a pas d'erreur lors du contrôle du total des engagements pour toutes les lignes de production, le système permet à l'utilisateur de visualiser le plan de pièces qu'il a élaboré et de choisir s'il valide ou non ce plan de pièces.

N.B. Le traitement d'erreur lors du contrôle du total des engagements n'a pas été pris en compte ici. On pourrait néanmoins envisager dans ce cas un retour au choix de lignes possibles, afin de permettre éventuellement à l'utilisateur d'élargir le nombre de lignes de production sur lesquelles on répartit les engagements.

5.4. FONCTIONS DE LA PHASE "REPARTITION DES ENGAGEMENTS"

REMARQUE PRELIMINAIRE

La découpe en fonctions de la phase "Répartition-des-engagements" est importante, dans la mesure où ce sont ces fonctions qui seront implémentées lors du prototypage.
Nous avons donc tenu à en donner un grand nombre d'informations. Après la dynamique des fonctions de la phase, on trouve une description précise de la statique de chaque fonction, s'appuyant sur le modèle de la boîte noire [BODART-83].

5.4.1. DYNAMIQUE

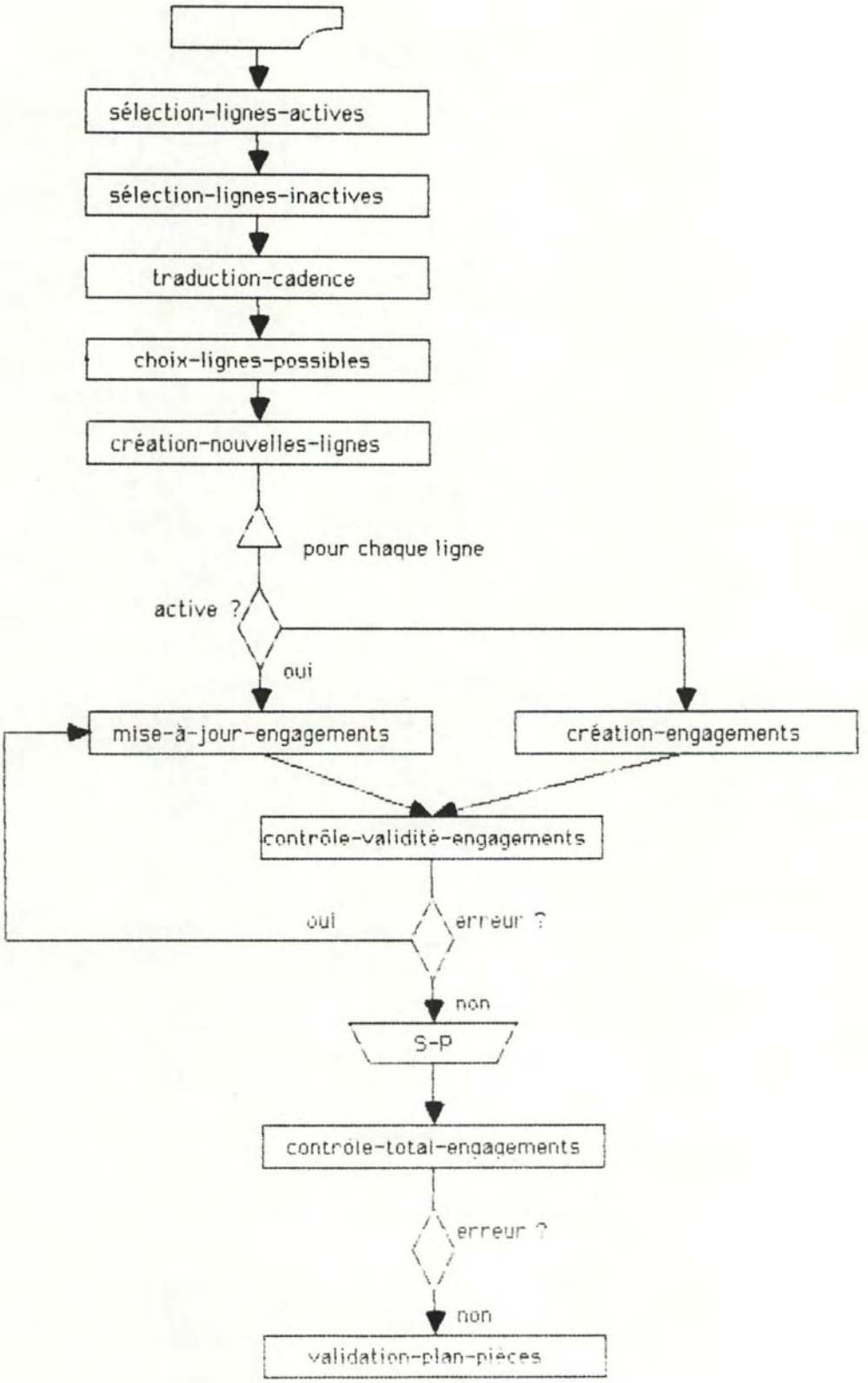


figure 5.2. Dynamique de la fonction "Répartition-des-engagements"

La dynamique des fonctions de la phase "Répartition-des-engagements" présentée à la figure 5.2. a été établie sans aucune étude approfondie de la meilleure solution informatique à implémenter.
Le seul critère pris en compte dans l'établissement de la dynamique a été de suivre le mieux possible les différentes étapes de réalisation de la répartition des engagements telles qu'elles existent actuellement, sans informatisation, aux Méthodes Mécaniques.
De plus, et ainsi que nous le verrons de façon approfondie dans la troisième partie, nous avons choisi une dynamique purement séquentielle, excluant la possibilité de choix par l'utilisateur d'un chemin à parcourir en fonction des traitements qu'il veut effectuer.
Quant à la structuration des fonctions, le principal critère d'identification est qu'à une fonction correspond une primitive d'action sur la mémoire du Système d'Information.

5.4.2. STATIQUE

NOTE : S.I. = Système d'Informations
T.E. = Type d'Entité
T.A. = Type d'Association

Fonction 1 : sélection-lignes-actives

- objectif : sélectionner dans la base de données toutes les lignes actives pouvant encore être engagées.
- message-donnée : plan de pièce à faire :
  - un organe ou une pièce
  - un pays
  - un code donnant le type (boîte, châssis, moteur, pièce).
- consultation-mémoire-S.I. : T.E. ligne-product
  T.A. montage-boîte-v,
  montage-châssis,
  montage-moteur,
  usinage.
- message-résultat : lignes actives, engagements des lignes actives.
- action-mémoire-S.I. : aucune.

Fonction 2 : sélection-lignes-inactives

- objectif : sélectionner dans la base de données toutes les lignes de production qui ne sont pas engagées.
- message-donnée : aucun.
- consultation-mémoire-S.I. : T.E. ligne-product,
  T.A. montage-boite-v,
  montage-chassis,
  montage-moteur,
  usinage.
- message-résultat : lignes inactives.
- action-mémoire-S.I. : aucune.

Fonction 3 : traduction-cadence

- objectif : rechercher les besoins correspondant à l'organe ou à la pièce et au pays donnés dans le plan de pièces à faire. Traduire ces besoins en cadences.
- message-donnée : plan de pièces à faire
- consultation-mémoire-S.I. : T.A. commerce-mot-DMC,
  commerce-bv-DMC,
  commerce-cha-DMC,
  commerce-pie-DMC.

- message-résultat : besoin (ensemble de 14 cadences correspondant au plan de pièces).
- action-mémoire-S. I. : aucune.

Fonction 4 : choix-lignes-possibles

- objectif : permettre à l'utilisateur de choisir, parmi les lignes actives et/ou inactives proposées, celles sur lesquelles il compte engager tout ou partie des besoins. Vérification que les lignes données par l'utilisateur existent bien dans les lignes actives et inactives.
- message-donnée : lignes actives ,
  lignes inactives,
  lignes choisies (par l'utilisateur).
- consultation-mémoire-S. I. : aucune.
- message-résultat : lignes choisies (correctes)
  et/ou messages d'erreur.
- action-mémoire-S. I. : aucune.

Fonction 5 : création-nouvelles-lignes

- objectif : permettre à l'utilisateur de créer de nouvelles lignes de production.
- message-donnée : aucun.
- consultation-mémoire-S. I. : aucune.
- message-résultat : lignes créées.
- action-mémoire-S. I. : aucune
  ou ajout d'un T. E. ligne-product.

Fonction 6 : mise-à-jour-engagements

- objectif : permettre à l'utilisateur de mettre à jour, pour une ligne de production active, un engagement pour chaque période, tout en visualisant les cadences requises. La mise-à-jour n'est pas effectuée à ce stade, il y a d'abord les contrôles.
- message-donnée : - une ligne de production active
  - plan de pièces à faire
  - besoin
- consultation-mémoire-S. I. : T. A. usinage,
  montage-boite-v,
  montage-chassis,
  montage-moteur.
- message-résultat : engagement (ensemble de 14 engagements pour la ligne et l'objet).
- action-mémoire-S. I. : aucune.

Fonction 7 : création-engagements

- objectif : permettre à l'utilisateur de donner, pour une ligne de production créée ou inactive, un engagement pour chaque période, tout en visualisant les cadences requises. La mise-à-jour n'est pas effectuée à ce stade, il y a d'abord les contrôles.
- message-donnée : - une ligne de production inactive ou créée
  - plan de pièces à faire
  - besoin
- consultation-mémoire-S. I. : aucune.
- message-résultat : engagement
- action-mémoire-S. I. : aucune.

Fonction 8 : contrôle-validité-engagement

- objectif : contrôler la validité des engagements d'une ligne de production introduits par l'utilisateur par rapport aux besoins et aux capacités.
- message-donnée : - une ligne de production active, inactive ou créée
  - engagement
  - besoin
- consultation-mémoire-S. I. : T. E. ligne-product.
- message-résultat : aucun
  ou messages d'erreur.
- action-mémoire-S. I. : aucune.

Fonction 9 : contrôle-total-engagements

- objectif : contrôler la validité des engagements introduits par l'utilisateur pour toutes les lignes de production choisies ou créées, par rapport aux besoins.
- message-donnée : - engagement (un par ligne de production).
  - besoin
  - lignes actives, inactives, créées.
- consultation-mémoire-S. I. : aucune.
- message-résultat : si erreur : lignes actives et inactives, messages d'erreur.
- action-mémoire-S. I. : aucune.

Fonction 10 : validation-plan-pièces

- objectif : permettre à l'utilisateur de visualiser le plan de pièces qu'il a élaboré, puis de choisir s'il le valide ou non.
- message-donnée : - besoin
  - plan de pièces à faire
  - lignes actives, inactives, créées.
- consultation-mémoire-S. I. : aucune.
- message-résultat : aucun.

- action-mémoire-S. I. : création ou modification de
  usinage,
  montage-boite-v,
  montage-chassis,
  montage-moteur.

5.4.3. L'AIDE A L'ELABORATION DES HYPOTHESES

Pour chaque fonction dont la statique est décrite au point 5.4.2., nous allons voir quels types d'aide à l'élaboration des hypothèses y sont implémentés.

Rappel : les requêtes passives peuvent être de
niveau 1 : les critères intervenant sont simples
niveau 2 : les critères intervenant sont liés à l'objet de sélection
niveau 3 : les critères intervenant sont croisés.
Ceci a été développé au paragraphe A. du point 2.3.3.

Fonction 1 : la sélection des lignes actives est une requête passive de niveau 3.

Fonction 2 : la sélection des lignes inactives est une requête passive de niveau 2.

Fonction 3 : afin de traduire les besoins en cadences, on a besoin d'une requête passive de niveau 1 pour trouver la relation désirée, puis l'aide se poursuit sous la forme de calcul de la cadence.

Fonction 4 : il n'y a aucune aide à l'élaboration des hypothèses dans le choix des lignes possibles.

Fonction 5 : idem pour la création des nouvelles lignes.

Fonction 6 : la mise-à-jour des engagements d'une ligne nécessite la recherche de la relation contenant les engagements (requête passive de niveau 2).

Fonction 7 : la création des engagements ne nécessite aucune aide à l'élaboration des hypothèses.

Fonction 8 : le contrôle de la validité des engagements d'une ligne utilise, outre une requête passive de niveau 1, une aide qui permet de respecter les contraintes imposées sur les engagements.

Fonction 9 : idem que pour fonction 8.

Fonction 10 : lors de la validation du plan de pièces, on a besoin d'une requête passive de niveau 2 pour créer ou modifier les relations de montage/usinage.

TROISIEME PARTIE : LE PROTOTYPAGE

INTRODUCTION

L'objectif de la troisième et dernière partie est de présenter la mise en oeuvre de l'application "plans de pièces" sur des outils de prototypage rapide : ORACLE et DSL-PROTO.

A cette fin, nous présentons, au chapitre 6, d'abord des définitions du prototypage et les principaux aspects de celui-ci: motivations, types, avantages. Ensuite, nous définissons les caractéristiques du prototypage de l'application "plans de pièces" : caractéristiques générales, caractéristiques de conception et caractéristiques d'utilisation. Nous terminons le chapitre en expliquant les contraintes concernant l'outil informatique. Ces contraintes seront utiles lors de la comparaison des deux outils de prototypage : ORACLE et DSL-PROTO.

Le chapitre 7 traite du logiciel ORACLE. Nous commençons par présenter ORACLE et ses outils de développement, ainsi qu'un aperçu des requêtes SQL. Puis nous énumérons les fonctions à réaliser. Nous explicitons ensuite les transformations nécessaires à effectuer par rapport aux résultats de l'analyse fonctionnelle. Ces transformations concernent les données et les traitements.
Ensuite, nous donnons l'implémentation d'une fonction avec ORACLE et nous terminons le chapitre par des commentaires sur l'utilisation de ce logiciel.

Le chapitre 8 traite du logiciel DSL-PROTO. Il reprend le même canevas que le chapitre 7.
Il présente DSL-PROTO, après en avoir défini le contexte. Puis les fonctions à réaliser sont énumérées. Nous développons ensuite les transformations. Elles sont de deux types : volontaires d'une part, nécessaires de par les fonctionnalités de DSL-PROTO d'autre part. Après avoir montré l'implémentation d'une fonction, nous développons les adaptations et les évolutions : limites dues à l'outil d'une part, problèmes méthodologiques d'autre part.

Le dernier chapitre (chapitre 9) est une comparaison des deux logiciels en tant qu'outils de prototypage de l'application "plans de pièces". Cette comparaison porte d'une part sur les fonctionnalités des deux logiciels, d'autre part sur les contraintes concernant l'outil informatique développées au chapitre 6.

CHAPITRE 6 : PROTOTYPAGE ; CONTRAINTES SPECIFIQUES DE MISE EN OEUVRE

INTRODUCTION

Dans ce chapitre, nous parlons des contraintes spécifiques de mise en oeuvre.

Ces contraintes se répartissent en deux catégories.

La première catégorie concerne les contraintes portant sur le prototypage en lui-même : quel type de prototypage, pour quels utilisateurs, dans quel cadre ? Avant de répondre à ces différentes questions, nous tentons de donner un aperçu des différentes définitions du prototypage.

La deuxième catégorie de contraintes porte plutôt sur l'outil informatique. Quelles sont les contraintes extérieures de mise en oeuvre ? Quels critères souhaités et/ou souhaitables peut-on définir ?

Dans ce but, nous présentons six contraintes spécifiques portant sur l'outil informatique. Elles concernent la souplesse, l'évolutivité, la simplicité, l'ergonomie, la modularité et la confidentialité.

6.1. DEFINITIONS ET ASPECTS DU PROTOTYPAGE

6.1.1. DEFINITIONS DU PROTOTYPAGE

Nous donnons ci-dessous quelques définitions du prototypage. Nous ne classons ni ne commentons ces définitions.

Néanmoins, nous pensons que la liste permet de donner un aperçu complet et valable de la notion de prototypage. Nous n'y relevons de plus aucune contradiction.

* Le prototypage est une maquette programmée du futur système, la maquette étant une réalisation à échelle réduite du système d'information projeté. [BOD-83] [PRO-86]

* Le prototypage est le "premier exemplaire d'un modèle construit préalablement à une fabrication de série." [RAU-86]

* Le prototypage est "une forme de maquette exécutable où l'accent est mis sur certains aspects du produit et où d'autres aspects sont ignorés, non décrits ou esquissés." [CHO-86]

* Le prototypage "sert à démontrer qu'une idée ou une théorie est utilisable et possède des débouchés... Un prototype doit pouvoir être utilisé pour résoudre des problèmes réels dans des conditions d'utilisation éventuellement restrictives." [DUC-86]

* "Le prototype d'un produit est un modèle qui sacrifie à la précision dans certains domaines pour permettre une vérification rapide des fonctions du produit." [CHO-86]

* "Le prototypage est une matérialisation incomplète du système à construire pour une partie ou la totalité de la spécification fonctionnelle." [LEG-2-86]

6.1.2. LES MOTIVATIONS DU PROTOTYPAGE

Outre les motivations découlant des définitions données au point 6.1.1., on peut relever 5 motivations importantes.

* "Depuis que le développement de l'informatique permet la réalisation et l'utilisation de logiciels de taille importante, le problème des erreurs conduisant à un mauvais fonctionnement... de ces logiciels est posé (...). Différentes tentatives ont été effectuées pour surmonter ce problème : il s'agit de la définition d'une programmation structurée, des travaux sur la spécification et les langages de spécification et, maintenant, le prototypage." [CHO-86]

* Le prototype est un logiciel visant à "introduire une approche plus expérimentale dans l'étude des problèmes d'organisation d'une part, en testant l'impact de diverses hypothèses de fonctionnement et d'autre part en favorisant la perception directe par les utilisateurs de ce qu'ils souhaitent obtenir." [BOD-83]

* "Le prototypage correspond à un besoin de vérifier le comportement réel du produit en cours de développement. Ce besoin montre que les méthodes de structuration, les langages de spécification, les outils divers les accompagnant ne permettent pas de saisir totalement tous les impacts d'un problème auquel on veut apporter un solution logicielle. Ils n'apportent pas de réponse à toutes les questions posées; une spécification est souvent difficile à lire, ou trop abstraite, et ne permet pas d'appréhender le comportement du produit spécifié dans son ensemble. Cela ne signifie pas pour autant qu'il faille abandonner ces solutions, mais plutôt les combiner avec le prototypage." [CHO-86]

* Les motivations du prototypage peuvent se refléter dans l'objet du prototype :
- prototypage de l'impact des modifications apportées au S.I. sur les performances du comportement de l'organisation, c'est-à-dire l'évaluation du caractère réalisable des spécifications.
- prototypage du poste de travail de l'utilisateur final, permettant aux utilisateurs de visualiser les écrans et d'expérimenter les dialogues qui seront mis à leur disposition dans le système réel.
- prototypage des règles de traitement de l'information, c'est-à-dire l'évaluation du caractère effectif des spécifications; celles-ci produisent les résultats attendus par les analystes et utilisateurs.
D'après [PRO-86].

* " Le besoin d'un outil de communication soit entre le spécifieur et le réalisateur, soit entre le vendeur et l'acheteur potentiel. Dans ce cas, le prototype est une approximation de l'interprétation que le réalisateur fait de la spécification... Le prototype permet donc d'obtenir rapidement des réactions sur la conception d'un produit, ..." [CHO-86]

6.1.3. LES TYPES DE PROTOTYPAGE

Les types de prototypage dépendent souvent des objectifs attendus.

[LEG-86] et [FLO-85] distinguent trois grands types de prototypage.

Il est clair que les outils informatiques de prototypage ne doivent pas être adaptés exclusivement à un type de prototypage.

1) Le prototype exploratoire

Les principales fonctions du prototype exploratoire sont :
- clarifier les besoins des utilisateurs
- aider à définir les principales caractéristiques du futur système.

Il permet le choix entre différentes solutions possibles et a donc un rôle essentiellement démonstratif.
Il s'agit d'un cahier des charges moderne, nécessitant une analyse préalable approfondie.

2) Le prototype d'expérimentation

Par développement incrémental, le prototype d'expérimentation permet de maîtriser la complexité et d'aider à vérifier la pertinence des solutions retenues.

Ses buts sont :
- étudier la faisabilité des parties critiques du système
- acquérir un savoir-faire sur le logiciel
- valider certaines options de conception.

Il s'agit d'un prototype orienté surtout sur les spécifications.

3) Le prototype pour évolution

Le prototype pour évolution est un produit fini directement utilisable, mais qui évolue de version en version.
Ce type de prototype
- est utilisé de manière opérationnelle
- doit répondre à des critères de qualité donnés par l'utilisateur
- doit permettre une réalisation rapide
- est surtout employé par l'utilisateur final.

6.1.4. AVANTAGES TIRES DU PROTOTYPAGE

D'après [CHO-86], les avantages tirés du prototypage (selon les expériences vécues), sont les suivants, outre ceux découlant des définitions citées en 6.1.1.

* Le prototypage conduit à des produits plus petits, de performance équivalente, avec moins d'efforts.
* Le prototypage permet d'obtenir un produit plus facile à maintenir.
* Le prototypage, par la meilleure communication en cours de conception et de réalisation qu'il offre, facilite une meilleure utilisation du produit par les utilisateurs.
* La précision et l'utilité du produit fini obtenu après prototypage semblent satisfaire davantage les utilisateurs que dans le cas d'un produit obtenu par les méthodes traditionnelles.
* Le prototypage est un excellent complément à la spécification, qu'il ne faut pas abandonner, car elle a l'avantage de produire des conceptions plus cohérentes et du logiciel plus facile à intégrer.

Et d'après [BOD-83],

" Faire prendre connaissance aux utilisateurs des spécifications d'un S.I., c'est-à-dire leur permettre de se représenter avec précision et de façon critique ce que produira le S.I. nouveau est un des principaux problèmes que rencontrent les concepteurs d'un S.I. "
Il existe de nombreuses solutions, classiques.

"La création de maquettes représente une autre solution à ce problème. Elle repose sur l'idée que la façon la plus immédiate et la plus significative pour un utilisateur de prendre connaissance des spécifications du S.I. nouveau est l'**expérimentation** directe, par ce dernier, de la solution proposée. A cette fin, on créera une maquette - ou prototype - programmée du S.I. futur."

6.2. LE PROTOTYPAGE DE L'APPLICATION "PLANS DE PIECES"

Nous donnons quelques caractéristiques du prototypage effectué pour l'application "Plans de pièces" du projet "Filière Méthodes Mécaniques".

On peut considérer qu'il y a trois catégories de caractéristiques :

- les caractéristiques générales
- les caractéristiques de conception
- les caractéristiques d'utilisation.

6.2.1. CARACTERISTIQUES GENERALES

Les caractéristiques générales du prototypage de l'application "Plans de pièces" situent le contexte général de réalisation de celui-ci.

* De façon générale, on peut considérer que le prototypage intervient, dans le temps, au niveau de l'analyse fonctionnelle, soit, suivant le cycle de vie d'un projet informatique, après l'étude d'opportunité ayant conduit à la décision de réaliser le projet.
Il n'en est pas de même pour le projet "Filière Méthodes Mécaniques". Le prototypage a été réalisé avant que le dossier pour investissements n'ait été signé.
Le prototype est donc un élément important dans la décision de vie du projet.

* De par sa nature, le système d'information n'est pas stable. De plus, la maîtrise des outils à mettre en oeuvre n'est pas suffisante pour faire du projet un environnement stable sur une période relativement longue (de l'ordre de dix ans).

* Le prototypage, vu l'étendue du projet et l'évolution technologique rapide, n'est applicable qu'à un "objet", à savoir un véhicule.
Le projet d'informatisation ne modifie en rien le schéma général d'industrialisation.

6.2.2. LES CARACTERISTIQUES DE CONCEPTION

Pour les concepteurs du projet, le prototypage représente un moyen de maîtriser la complexité et d'aider à vérifier la pertinence des solutions retenues.
Le prototypage doit rester très proche des spécifications.
On se trouve donc bien en présence d'un prototype d'expérimentation.

En outre, il est important de souligner le phénomène suivant.
Les concepts utilisés et les traitements réalisés dans le S.I. sont très difficiles non seulement à percevoir, mais aussi à modéliser.
De plus, il existe une certaine difficulté d'acquisition et de formalisation des connaissances. L'information est difficile à obtenir. Venant de diverses sources, elle est souvent contradictoire. Pour terminer la lithanie, ajoutons que pour des raisons "politiques" ou de confidentialité, beaucoup d'informations sont "omises", verbalement ou par écrit, même de façon interne.
Aussi le prototypage est-il effectué à partir de spécifications tout à fait incomplètes, parfois même contradictoires. Le prototypage est par conséquent pour les concepteurs **aussi** un moyen d'en savoir plus sur le projet informatique à réaliser, de par les dialogues avec les utilisateurs, étape importante dans la réalisation d'un prototypage.

6.2.3. LES CARACTERISTIQUES D'UTILISATION

L'impression principale qui ressort des dialogues avec les utilisateurs finals du projet informatique est qu'ils se désintéressent totalement de l'aspect spécifications, faisabilité, etc...
On retrouve dans leur conception du prototypage toutes les caractéristiques du prototype pour évolution. Ce qui intéresse surtout les utilisateurs finals, c'est la cinématique des écrans, l'ergonomie, la simplicité, la facilité et le soulagement de tâches manuelles ardues, longues et répétitives. Leurs objectifs se situent donc à un autre niveau que ceux de la Direction, que nous avons développé au chapitre 1.

Nous verrons dans la suite du mémoire comment les outils utilisés pour prototyper parviennent à concilier la juxtaposition de ces deux types de prototypage.

6.3. CONTRAINTES CONCERNANT L'OUTIL INFORMATIQUE

Outre les contraintes spécifiques de mise en oeuvre qui découlent implicitement des caractéristiques vues au point 6.2., il existe des contraintes spécifiques de mise en oeuvre qui concernent l'outil informatique. Nous en énumérons et commentons quelques-unes.

6.3.1. SIMPLICITE

Un des principaux buts du prototypage de l'application "Plans de pièces" est la simplicité d'utilisation et de réalisation, étant donné les compétences réduites et la faible propension à apprendre des utilisateurs finals.
L'outil informatique doit donc offrir le meilleur niveau possible d'accessibilité aux non-informaticiens.

6.3.2. EVOLUTIVITE

L'évolutivité découle naturellement de la typologie de prototypage souhaitée par les utilisateurs finals, à savoir un prototypage évoluant de version en version.
L'évolutivité se manifeste aussi dans le besoin qu'ont les concepteurs de se servir du prototypage comme moyen d'acquisition et de consolidation des spécifications. (cfr 6.2.2.)

6.3.3. MODULARITE

Le prototypage de l'application "Plans de pièces" doit pouvoir être intégré à un prototypage plus vaste, celui du projet "Filière Méthodes Mécaniques", ceci du point de vue des concepteurs.
Quant aux utilisateurs, ils ont un objectif de consolidation futur du prototype avec le noyau technique.

6.3.4. CONFIDENTIALITE

Un type d'industrie comme la construction automobile, par sa technologie de pointe très évolutive et par sa concurrence importante, exige un niveau de confidentialité très élevé, que ce soit de façon externe ou, ce qui nous intéresse, de façon interne.

Dans la possibilité d'intégration du prototype de l'application "Plans de pièces" à d'autres modules, les utilisateurs finals tiennent à ce que non seulement leurs techniques de travail, mais aussi et surtout les données sur lesquelles ils travaillent, restent dans leur espace privé. Or, il est clair que dans ce type de prototypage, on doit travailler sur des objets existants, afin que l'utilisateur s'y retrouve et qu'il se familiarise avec le nouveau produit.

6.3.5. ERGONOMIE

L'ergonomie est très importante.
Rappelons-nous que le prototype est un élément de décision de continuation du projet.
D'autre part, nous avons vu que les utilisateurs finals considèrent ce prototypage comme la version intermédiaire d'un produit fini. Le prototype doit donc les satisfaire sur le plan des "conditions de travail".

6.3.6. SOUPLESSE

La souplesse du prototype est une contrainte qui découle des autres.
Pour être simple, évolutif, modulaire, ..., l'outil de prototypage doit être souple.

CHAPITRE 7 : LE LOGICIEL ORACLE : PRESENTATION ET FONCTIONS REALISEES

INTRODUCTION

Le premier objectif de ce septième chapitre est de présenter brièvement le logiciel ORACLE, qui sera comparé plus loin dans le travail avec le logiciel DSL-PROTO, en tant qu'outils de prototypage de l'application "Plans de Pièces" du projet "Filière Méthodes Mécaniques".

Le second objectif est de décrire les fonctions réalisées avec le logiciel ORACLE.

Le chapitre contient d'abord une énumération des fonctions à réaliser, qui proviennent de la découpe de la phase "Répartition-des-engagements" de l'application "Plans de pièces". On y trouve également le sous-schéma conceptuel des données de cette phase.

Ensuite, sont expliquées les transformations nécessaires pour implémenter avec ORACLE ces fonctions. Les transformations sont de deux types : concernant les données et concernant les traitements.

Puis nous décrivons l'implémentation d'une partie de la phase. L'implémentation complète se trouve à l'annexe 3.

Nous terminons le chapitre en critiquant la version d'ORACLE utilisée.

7.1. PRESENTATION D'ORACLE

7.1.1. PRESENTATION GENERALE

ORACLE est un Système de Gestion de Bases de Données (S.G.B.D.) relationnel. Il est développé par ORACLE CORPORATION.

ORACLE utilise un langage relationnel, SQL Plus. Il s'agit d'une extension de SQL/DS d'IBM, avec lequel il est compatible.

Avec SQL Plus, ORACLE CORPORATION a étendu les possibilités de SQL et l'a rendu plus proche de l'utilisateur.

Il s'agit d'un langage non procédural.

ORACLE est doté d'un ensemble d'**outils d'aide au développement**. Outre le langage SQL Plus, on retrouve un générateur d'état interactif, un dictionnaire de données intégré, un générateur d'application, un gestionnaire d'écran, et un système de préparation de documentation.

ORACLE étant un S. G. B. D. de type relationnel, les données se présentent sous forme de tables. Chaque table est composée d'enregistrements contenant plusieurs colonnes.
La composition de chaque table est connue du dictionnaire de données ORACLE.

Les données sont manipulables à l'aide d'un langage basé sur la logique de l'algèbre relationnelle (description du résultat attendu) (SQL).

Le lecteur désirant approfondir l'étude du logiciel ORACLE peut consulter entre autres [REN-2-85], [ORA-1-84], [ORA-2-84], [ORA-3-84].

7.1.2. LES OUTILS DE DEVELOPPEMENT D'ORACLE

La figure 7.1. présente le développement d'une application ORACLE.

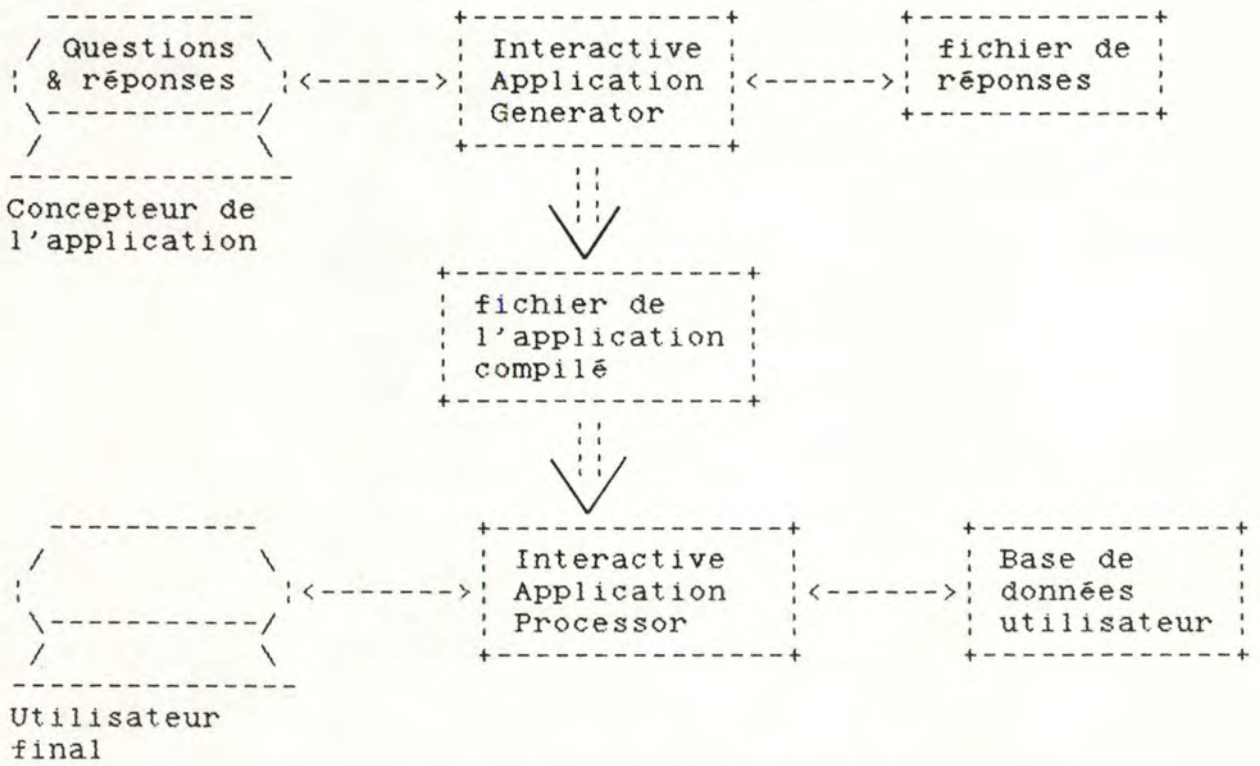


figure 7.1. le développement d'une application ORACLE

Les outils de développement d'ORACLE sont contenus dans le IAF (Interactive Application Facility).
IAF permet de créer une application interactivement, par un dialogue questions-réponses avec le concepteur de l'application.
IAF utilise à cette fin IAG (Interactive Application Generator), qui génère, à partir du fichier de réponses un fichier de l'application compilé et utilisable par IAP (Interactive Application Processor).
IAP permet à l'utilisateur final d'exécuter l'application avec les données de la base.

Nous allons présenter les deux composants d'IAF, à savoir :
- IAG, le générateur interactif d'application
- IAP, le processeur interactif d'application.

A. LE GENERATEUR INTERACTIF D'APPLICATION (IAG)

Le grand principe est qu'il y a un dialogue au terminal entre IAG et le concepteur de l'application. Ce dernier doit répondre à des questions concernant l'application.
Les questions appartiennent à quatre grandes catégories.

1) questions générales sur l'exécution de l'application.
2) questions sur les tables référencées.
3) questions sur les champs (formats, domaines de valeurs, manipulations,...).
4) questions sur le texte libre et le graphisme des écrans.

Précisons quelques notions propres à une application ORACLE.
On peut dire qu'en règle générale :

- Une application est composée de plusieurs blocs.
- Un bloc est une section de l'application qui permet à l'utilisateur d'insérer, de modifier et de détruire des informations ainsi que de poser des questions à leur sujet, pour **une table**.
  Un bloc contient un ou plusieurs champs de la table spécifiée et peut contenir des **références** à des champs d'autres tables.
  Un bloc peut afficher une ou plusieurs occurrences des champs spécifiés.
- Un champ est une colonne d'une table.
  Les champs affichés peuvent ne pas appartenir à la table.
  Les champs peuvent éventuellement avoir une valeur recopiée d'un champ d'un autre bloc ou une valeur par défaut.
  Une ou plusieurs occurrences d'un même champ peuvent être affichées.

Chaque champ peut contenir des ordres SQL. Ceux-ci sont exécutés lors de l'insertion, la modification, la suppression ou la recherche d'articles. Nous présentons les ordres SQL au point 7.1.3.

B. LE PROCESSEUR INTERACTIF D'APPLICATION (IAP)

A partir du fichier compilé par IAG, le processeur interactif d'application (IAP) exécute l'application.
IAP
- lit la forme compilée
- accède à la base de données
- interagit avec l'utilisateur final au terminal.

IAP peut être utilisé pour entrer des informations dans une base de données ORACLE ou pour exécuter une requête pour sélectionner et afficher des informations venant de la base de données.
L'information affichée peut être changée ou enlevée de la base.
IAP comprend toutes sortes de contrôles qui peuvent être exécutés sur les données entrées de telle façon que le contenu de la base de données soit exempt d'erreur.
IAP peut aussi limiter les actions que les utilisateurs sont autorisés à accomplir.
Les fonctions IAP sont exécutées en réponse à des commandes venant du terminal.

Les fonctions IAP peuvent être divisées en deux catégories :
les fonctions de gestion d'écran
et les fonctions de manipulation des données (query, update, insert, delete, query where <critère>).

7.1.3. LES REQUETES SQL

Nous avons déjà parlé du langage SQL, ainsi que de son utilité dans le cadre de la génération d'une application ORACLE.

Globalement, on peut dire que les requêtes SQL permettent de :
- retirer de l'information de la base de données et l'afficher
- faire référence à des valeurs de champs affichés à l'écran, valeurs soit entrées par l'utilisateur, soit extraites de la base de données par un QUERY
- valider les données introduites ou modifiées pour maintenir la consistance et l'intégrité de la base de données.

Lors de la génération de l'application par le concepteur, les requêtes SQL peuvent intervenir à deux niveaux :

- requêtes SQL de niveau champ : elles sont exécutées lorsque pour un champ, on entre, modifie, ou enlève une valeur ou pour un champ d'une ligne satisfaisant un QUERY.
Une ou plusieurs requêtes SQL peuvent être spécifiées par champ.
Ces requêtes sont SELECT, UPDATE, INSERT, DELETE,...
Elles permettent l'affichage des données, l'établissement de valeurs par défaut, les calculs sur les données, la vérification des valeurs de champs introduites, la vérification du format, du rang, du domaine de valeurs d'un champ.

- requêtes SQL de niveau fonction : elles sont exécutées lorsque l'utilisateur fait une demande de traitement de transaction en invoquant la fonction "COMMIT" (fin de transaction) ou la fonction "EXECUTE QUERY" (exécution d'une recherche), c'est-à-dire lors du transfert de données dans un sens ou dans l'autre entre l'écran IAP et la base de données.
Ces requêtes sont appelées "**TRIGGERS**" car elles sont déclenchées par l'action de l'opérateur.
Les TRIGGERS sont associés à un bloc, contrairement aux requêtes SQL de niveau champ.
Les TRIGGERS peuvent effectuer les opérations suivantes :
QUERY (recherche)
INSERT (insertion)
DELETE (suppression)
UPDATE (mise-à-jour).

7.2. FONCTIONS A REALISER

Les fonctions à réaliser sont celles qui proviennent de la découpe de la phase "Répartition-des-engagements" de l'application "Plan de pièces".

Nous énumérons ci-dessous ces fonctions. Leur dynamique et leur statique ont été décrites aux points 5.4.1 et 5.4.2.

Fonction 1 : sélection-lignes-actives
Fonction 2 : sélection-lignes-inactives
Fonction 3 : traduction-cadence
Fonction 4 : choix-lignes-possibles
Fonction 5 : création-nouvelles-lignes
Fonction 6 : mise-à-jour-engagements
Fonction 7 : création-engagements
Fonction 8 : contrôle-validité-engagements
Fonction 9 : contrôle-total-engagements
Fonction 10 : validation-plan-pièces

La figure 7.2. (page suivante) présente le sous-schéma conceptuel des données de la phase "Répartition-des-engagements".

7.3. LES TRANSFORMATIONS NECESSAIRES

A partir des résultats de l'analyse fonctionnelle de la phase "Répartition-des-engagements" de l'application "Plan de pièces", à savoir la structuration des données, la structuration, la dynamique et la statique des traitements, certaines transformations sont nécessaires pour réaliser les fonctions grâce au S.G.B.D. relationnel ORACLE.

Ces transformations sont de deux types, que nous détaillons ci-dessous.

1) Transformations du schéma conceptuel des données : il s'agit de passer du modèle Entité-Association au modèle relationnel. (cfr point 7.3.1.)

2) Transformations des traitements : il s'agit d'orienter la découpe des traitements autrement que selon les principes utilisés dans la méthodologie d'analyse fonctionnelle développée dans [BOD-83]. (cfr point 7.3.2.)

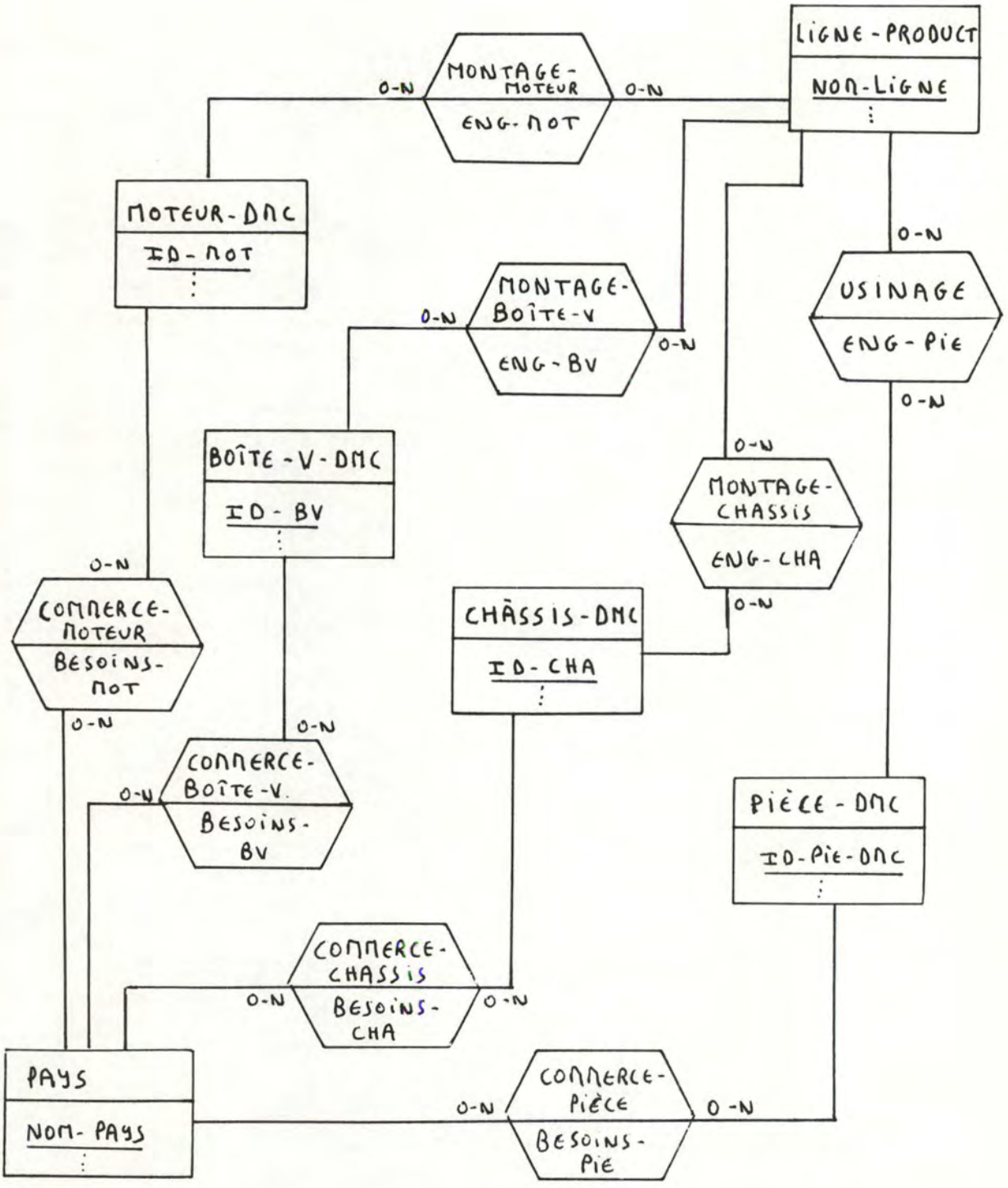


figure 7.2. Le sous-schéma conceptuel des données de la phase "Répartition-des-engagements"

7.3.1. PASSAGE DU MODELE E-A AU MODELE RELATIONNEL

ORACLE étant un S.G.B.D. relationnel travaille sur des tables. Il est donc nécessaire de transformer les composants du modèle Entité-Association en tables relationnelles.

Il existe toute une méthodologie de transformation d'un schéma entité-association en schéma relationnel. Cette méthodologie est abondamment illustrée dans la littérature. On peut notamment consulter [HAI-86].

Dans notre démarche concernant le prototypage de l'application "Plan de pièces", il y a une volonté de ne pas optimiser ce processus de transformation.
On peut y voir deux raisons :
1) Il s'agit d'un prototype : on travaille donc sur un ensemble réduit de données et les problèmes de facilité d'accès et de temps d'accès sont écartés.
2) Vu le but de comparaison de deux outils de prototypage, il faut garder une échelle de comparaison. Nous n'avons pas voulu biaiser au départ la comparaison en partant de schémas de structuration des données trop différents.
Nous avons utilisé trois grands principes de transformation, auxquels nous avons ensuite apporté quelques simplifications.

Principes de transformation

1) Une entité est représentée par une table identifiée par l'identifiant de l'entité.
2) Une relation est représentée par une table identifiée par les identifiants des entités que la relation relie.
3) Un groupe est représenté par un ensemble d'attributs.

Par conséquent, à partir du sous-schéma conceptuel de la phase "Répartition-des-engagements" (figure 7.2.), on obtient les tables relationnelles suivantes (on trouve le nom de la table suivi de ses attributs entre parenthèses ; l'identifiant est souligné) :

* pie-m (id-pie-m)
* pays (nom-pays)
* mot-m (id-mot-m)
* cha-m (id-cha-m)
* bv-m (id-bv-m)
* ligne (nom-ligne, capacité, p1-ech, p2-ech, ..., p14-ech)
* comm-m (id-m, nom-pays, type-m, p1-besoins-m, p2-besoins-m, ..., p14-besoins-m)
* montusi (id-m, nom-ligne, type-m, p1-engage, p2-engage, ..., p14-engage)

Les simplifications apportées par rapport aux principes de transformation énumérés consistent à :

1) regrouper certaines tables pour simplifier la gestion de celles-ci.
   Exemple : les 4 tables "commerce-xx-dmc" deviennent la seule table "comm-m" sous réserve de l'ajout d'un attribut spécifiant le type d'objet (moteur, boîte de vitesses, châssis, pièce) qui est commercialisé dans le pays.

2) avoir une dénomination des objets plus courte (i.e. plus adaptée à ORACLE).

7.3.2. ARCHITECTURE DES TRAITEMENTS

Nous avons décrit au point 7.1.2. les différents aspects de la génération d'une application avec le logiciel ORACLE.
Rappelons qu'une application est composée de blocs. A chaque bloc est associée une table. La notion d'écran associé à un bloc est très importante.

C'est à partir de ces notions de blocs, tables et écrans que nous devons découper les traitements à implémenter sous ORACLE.

Nous montrons ci-dessous les correspondances et les éventuelles variations dans les deux découpes.

BLOC 0 : TRADUCTION

Dans le bloc 0, TRADUCTION, on trouve notamment la sélection du plan de pièces à faire. Il s'agit donc des traitements en amont de la phase "Répartition-des-engagements". Un résultat de ces traitements correspond au message initial de la phase.

Nous citons ce bloc ici, car à l'intérieur de celui-ci, a été implémentée la traduction des besoins en cadences, qui correspond à la fonction 3.
Cela a été fait à ce niveau pour des raisons de simplification de génération de l'application (définitions des champs, requêtes, etc...) et aurait pu être fait plus tard dans le déroulement de la phase.

C'est ce qui justifie l'appellation "BLOC 0".

BLOC 1 : ENGAGE1

Le bloc 1, ENGAGE1, travaille sur la table "ligne". Il correspond au premier type d'écran (écran 1).
Dans ce bloc, on réalise quatre fonctions :
fonction 2 : sélection-lignes-inactives
fonction 4 : choix-lignes-possibles
fonction 7 : création-engagements
fonction 8 : contrôle-validité-engagements

En réalité, on effectue tous les traitements associés aux lignes inactives, à savoir leur sélection dans la base de données, le choix de certaines d'entre elles par l'utilisateur, la création des engagements sur les lignes choisies et le contrôle de la validité des engagements introduits par l'utilisateur.
On travaille donc sur **l'écran** des lignes inactives.

La figure 7.3. (écran 1/2) montre l'écran après sélection des lignes inactives (que l'utilisateur réalise par la touche-fonction "QUERY").
Le lecteur peut trouver en annexe 2 les différents écrans ORACLE.

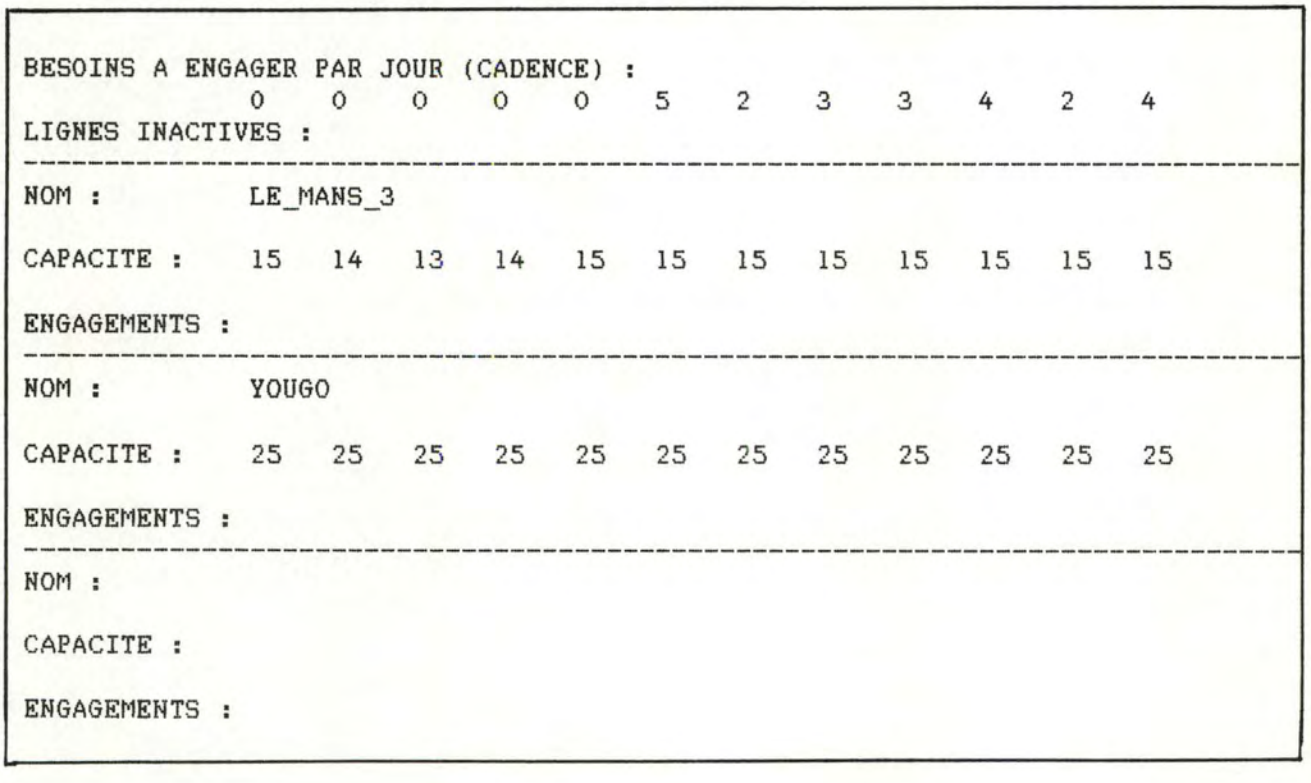


figure 7.3. Exemple d'écran ORACLE

BLOC 2 : ENGAGE2

Le bloc 2, ENGAGE2, travaille sur la table "montusi". Il correspond au deuxième type d'écran (écran 2).
Dans ce bloc, on réalise quatre fonctions :
fonction 1 : sélection-lignes-actives
fonction 4 : choix-lignes-possibles
fonction 6 : mise-à-jour-engagements
fonction 8 : contrôle-validité-engagements

Le principe des traitements est identique à celui du bloc 1.

BLOC 3 : ENGAGE3

Le bloc 3, ENGAGE3 travaille sur la table "ligne". Il correspond au troisième type d'écran (écran 3).
Dans ce bloc, on réalise trois fonctions :
fonction 5 : création-nouvelles-lignes
fonction 7 : création-engagements
fonction 8 : contrôle-validité-engagements

Le principe des traitements est identique à celui du bloc 1.

BLOC 4 : CONTROLE

Le bloc 4 est un bloc de contrôle. Aucune fonction n'y est réalisée.
Le bloc de contrôle est un artifice employé par le concepteur afin de gérer des liens complexes entre blocs. Nous n'y attachons pas d'importance ici.

BLOC 5 : AFFICHAGE-PLAN

Le bloc 5, AFFICHAGE-PLAN, travaille sur la table "montusi". Il correspond au quatrième type d'écran (écran 4).
Dans ce bloc, on réalise une seule fonction :
fonction 10 : validation-plan-pieces

N.B. : la fonction 9 (contrôle-total-engagements) n'a pas été implémentée. Ce n'est ni un oubli ni une impossibilité, mais simplement une simplification des traitements exigée par des contraintes de temps.

CONCLUSION

On constate donc que la découpe des traitements est différente puisque les traitements dans ORACLE sont "orientés-écran".
Néanmoins, si la forme varie, le contenu ne change pas. On a donc réalisé en fin de parcours, les mêmes traitements, en tout cas au point de vue dynamique.
La différence fondamentale est qu'on exécute la dynamique découlant de l'analyse fonctionnelle **une fois** par type d'écran.

Remarque : l'utilisateur peut néanmoins, s'il le désire, presque respecter la dynamique de l'analyse fonctionnelle puisque, après avoir sélectionné les lignes inactives par exemple, il peut passer au bloc suivant (touche-fonction NEXT BLOCK) et sélectionner les lignes actives.

Nous approfondissons bien sûr toutes ces constatations lors de la comparaison.

On remarque que dans les traitements réalisés par ORACLE, certains choix de l'utilisateur (engagements, lignes choisies, etc...) sont directement enregistrés dans la base de données, contrairement à ce qui est écrit dans la description de la statique de la phase "Répartition-des-engagements" où certaiøns changements de l'état de la base de données sont effectués après l'accord de validation du plan de pièces par l'utilisateur.
Ceci est dû au simple fait que nous avons géré, dans le développement de la phase avec ORACLE, le numéro d'hypothèse, contrairement au développement avec DSL-PROTO. On peut donc se permettre, dans le traitement ORACLE, de modifier l'état de la base de données, puisqu'un numéro est enregistré comme témoin d'une hypothèse de travail à éventuellement valider ultérieurement.

Signalons aussi que les liens entre les différents blocs sont développés plus loin dans le chapitre.

7.4. IMPLEMENTATION

Nous montrons dans ce point l'implémentation du bloc 1 : engage1.
L'annexe 3 contient l'implémentation de tous les blocs, ainsi que l'implémentation des aides à l'élaboration des hypothèses.
L'annexe 5 contient le code d'une partie du bloc d'affichage du plan de pièces.

BLOC 1 : ENGAGE1

1) Lorsque l'utilisateur visualise l'écran 1/1 (annexe 2), il faut qu'il puisse, par la touche-fonction QUERY, consulter les lignes inactives.
Réalisation : par une clause WHERE par défaut, qui sera automatiquement exécutée lorsque l'utilisateur fera le QUERY.

```
; Block name :
engage1
; Enter default WHERE and ORDER BY clause :
WHERE nom-ligne NOT IN (SELECT nom-ligne FROM montusi)
```

2) Pour choisir la ligne sur laquelle il désire travailler, l'utilisateur positionne simplement le curseur sur l'écran.

3) La création des engagements consiste en une introduction par l'utilisateur de ceux-ci.
Pour chaque valeur entrée, les contrôles suivants sont effectués:
les contraintes besoin >= engagement
capacité >= engagement doivent être respectées.

Soient . le champ p1-engage contenant l'engagement introduit par l'utilisateur
. le champ af1-p1-cadence contenant la cadence requise (calculée, pour rappel, dans le bloc 0)
. le champ p1-ech contenant la capacité de la ligne.

Réalisation :

```
; Field name :
p1-engage
; ...
.
.
; SQL >
SELECT *
FROM DUMMY
WHERE (:engage1.p1-engage <= :traduction.af1-p1-cadence)
```

```
/
; Message if value not found :
Engagement > besoin
; Must value exist Y/N :
Y
SELECT *
FROM DUMMY
WHERE (:engage1.p1-engage <= :engage1.p1-ech)

; Message if value not found :
Engagement > capacité
; Must value exist Y/N :
Y
```

La combinaison de ces questions-réponses oblige l'utilisateur à ne rentrer que des engagements respectant les contraintes implémentées.

4) La création physique des engagements dans la table adéquate, "montusi", est effectuée par le TRIGGERS pré-UPDATE auquel on associe l'ordre SQL INSERT.

Réalisation :

```
; Field name :
PRE-UPDATE
; SQL >
INSERT INTO montusi VALUES (:traduction.id-m,
                            :traduction.type-m,
                            :engage1.nom-ligne,
                            :engage1.p1-engage,
                            :engage1.p2-engage,
                            ...
                            :engage1.p14-engage)
```

Remarque : On peut considérer que le lien entre le bloc 0 et le bloc 1 est réalisé par l'affichage sur le bloc 1 de champs dont les valeurs sont calculées dans le bloc 0 (les cadences).
Il est aussi réalisé lors du TRIGGERS, puisque l'insertion d'une ligne dans une table exige que la ligne soit complète. Il faut donc aller rechercher dans le bloc TRADUCTION les valeurs de id-m et type-m.

7.5. COMMENTAIRES

Les commentaires concernant le logiciel ORACLE ont pour objet des critiques quant à l'utilisation de celui-ci et aux extensions envisagées.

1. GENERATION DE L'APPLICATION

Lors du dialogue entre IAG et le concepteur de l'application, il n'y a pas moyen de revenir en arrière.
La moindre erreur ne peut donc pas être corrigée interactivement. Elle peut de plus entraîner l'obligation de répondre à une série de questions auxquelles le concepteur ne s'attend pas à et ne désire pas répondre. C'est un inconvénient.
Néanmoins, tous les couples questions-réponses sont sauvés dans un fichier que le concepteur peut, après le dialogue, modifier grâce à un éditeur de texte.
Dans ce cas, le concepteur de l'application doit être prudent lorsqu'il modifie une réponse car la séquentialité des questions peut être modifiée en fonction de la réponse.
Aussi, lors de la recompilation du fichier après modification, d'autres erreurs peuvent surgir, IAG prenant les réponses de façon séquentielle.

Exemple :

(les ; en début de ligne signifient : questions de IAG)

1) Fichier après dialogue interactif :

```
; Display prompt above field Y/N :
Y
; Allow field to be entered :
N
; Execute a SQL statement :
SQL >
<return>
; ...
```

La séquence de réponses est : Y, N, <return>.

2) L'utilisateur modifie, en éditeur, la deuxième réponse.

La séquence de réponses devient : Y, Y, <return>.

3) A la recompilation, IAG, lorsqu'il reçoit une réponse Y à la question "Allow field to be entered", pose la question

; Allow field to be updated :

et attend une réponse Y ou N. Or, il prend la réponse suivante, qui est ici <return> et il y a par conséquent erreur.

Extension envisagée : disposer d'un générateur d'application travaillant par menus et par cadres sur les différents types de questions.

2. LA TECHNIQUE DES "SWITCHES"

Une amélioration à la technique que nous venons de voir consiste à travailler avec des "switches".
Dans l'exemple développé ci-dessus, avant de pouvoir répondre à d'éventuelles questions supplémentaires, l'utilisateur, en éditeur, modifie sa réponse à la deuxième question en changeant "N" par "%SW". A la recompilation, IAG rencontre ce switch et reprend alors le dialogue en interactif. L'utilisateur peut alors répondre aux éventuelles questions supplémentaires et donner l'ordre de repasser aux réponses déjà contenues dans le fichier en répondant "%SW" à la première question non supplémentaire.

3. RAPPORT ENTRE BLOC ET ECRAN

En toute généralité, on peut considérer qu'à un bloc correspond un écran au terminal.

Exemple :

Sur un premier écran, l'utilisateur travaille (c'est-à-dire consulte, modifie, fait des sélections, ...) sur la table "EMPLOYE".
Sur un deuxième écran, l'utilisateur travaille sur la table "PROJET".
Il est clair qu'il doit exister normalement un lien entre les deux écrans, entre les deux blocs (par exemple l'affichage des projets de l'employé sélectionné).
Pour créer le lien, le concepteur de l'application peut demander que dans un bloc (i), les valeurs affichées appartiennent à la même ligne de table que celle dont un champ (affiché ou non affiché) dans le bloc (i) a une valeur copiée d'un bloc en amont (i-1).
On a donc un lien, une valeur commune entre des blocs. On peut donc gérer une forme de dynamique entre blocs.

4. LA GENERATION RAPIDE

Lorsque le concepteur de l'application génère une longue application, travaillant sur beaucoup de tables contenant beaucoup de champs, le dialogue interactif géré par IAG peut paraitre fastidieux, long et être source d'erreurs.
Aussi, existe-t'il une technique permettant de créer **rapidement** une application : FASTFORM.
FASTFORM génère une application basée sur les définitions des tables du dictionnaire de données. L'application générée par FASTFORM est évidemment moins complète, moins détaillée que celle générée classiquement, et toutes les questions concernant l'affichage, le positionnement, etc... sont omises.

5. REPETITIVITE DANS LA GENERATION

Nous disions plus haut que la création d'une application par le concepteur pouvait être longue et fastidieuse. En voici une autre preuve.
- Si un champ est repris dans plusieurs blocs (faisant partie ou non de la table associée au bloc), il faut le définir autant de fois qu'il est repris, et répondre chaque fois aux questions y associées.
- De même, si un champ est affiché à plusieurs endroits dans un même bloc, il faut répéter le même processus avec des noms de champ différents, puisqu'à une déclaration de champ est associée une seule position physique sur une page d'écran.

CHAPITRE 8 : LE LOGICIEL DSL-PROTO : PRESENTATION ET FONCTIONS REALISEES

INTRODUCTION

Le premier but de ce chapitre est de présenter le logiciel DSL-PROTO, qui est un outil faisant partie de l'atelier-logiciel IDA (Interactive Design Approach) réalisé à l'Institut d'Informatique de Namur.

Le second but du chapitre 8 est de décrire les fonctions réalisées avec le logiciel DSL-PROTO.

Nous commencons par une énumération des fonctions à réaliser, qui proviennent de la découpe de la phase "Répartition-des-engagements" de l'application "Plan de pièces".

Nous expliquons ensuite les transformations nécessaires pour implémenter avec DSL-PROTO ces fonctions.

8.1. LE CONTEXTE DE DSL-PROTO

8.1.1. PRESENTATION GENERALE

DSL-PROTO est un des outils faisant partie du système-logiciel IDA (Interactive Design Approach) réalisé à l'Institut d'Informatique des Facultés Universitaires Notre-Dame de la Paix à Namur.

IDA est développé en coopération avec le projet ISDOS développé à l'université de Michigan.

Les buts principaux d'IDA sont de permettre au concepteur de s'assurer :

- que les spécifications qu'il reçoit sont suffisantes
- que le système conçu est faisable compte tenu des contraintes qui lui sont imposées
- que la logique interne de son produit est correcte.

Le lecteur désirant plus d'informations au sujet du logiciel IDA consultera [BOD-83], [BLI-86], [DSA-84].

8.1.2. ARCHITECTURE DU LOGICIEL IDA

IDA est structuré de la façon suivante :
le coeur du système est la base de données des spécifications du S.I., décrites en langage DSL (Dynamic Specification Language).

Cette base de données est la source unique de toute la documentation.

IDA comprend trois outils
- l'analyseur DSA (Dynamic System Analyser) (appelé depuis peu SPEC) (cfr point 8.1.3.).
- le générateur automatique de programmes de simulation pour évaluer le caractère **réalisable** du système décrit : DSL-SIM (cfr point 8.1.4.).
- le générateur d'une maquette programmée du système futur pour tester le caractère **effectif** des spécifications : DSL-PROTO (cfr point 8.2.).

8.1.3. DSA (SPEC)

DSA (Dynamic Systems Analyser) "désigne l'ensemble des programmes qui exploitent la base de données de spécifications." [BOD-83]

DSA possède son propre analyseur de commandes.

Les opérations sur la base de données permises grâce à DSA sont :
- la mise à jour
- le langage d'interrogation de la base de données (Query Language). L'interrogation se fait à partir de critères complexes
- la création de rapports documentaires à destination du concepteur et du client, permettant "d'extraire de la base de données, de manière complète, partielle ou synthétique, les descriptions, pour les présenter de manières variées (...) et sur des périphériques divers (...)." [BOD-83]
  Il existe des rapports effectuant des contrôles de cohérence et de complétude des descriptions introduites.

8.1.4. DSL-SIM

On peut évaluer le caractère réalisable des spécifications par génération automatique d'un programme de simulation DSL-SIM.

"Les spécifications conceptuelles sont indépendantes des ressources humaines, matérielles, financières ou organisationnelles nécessaires à leur mise en oeuvre. Il importe, dès lors, d'évaluer si une solution conceptuelle est **réalisable** par rapport aux **ressources** dont dispose l'organisation : effectifs en personnel, définition des postes et des horaires de travail, nombre de terminaux à mettre en oeuvre, organisation des circuits administratifs et des canaux de communication des informations, compatibilité entre traitements manuels et traitements automatisés, ...
Pour évaluer le caractère réalisable de tout ou partie du S.I. on se basera uniquement sur les spécifications dynamiques du S.I. et sur les spécifications des ressources." [BOD-83]

8.2. PRESENTATION GENERALE DE DSL-PROTO

8.2.1. DEFINITION

DSL-PROTO est un des trois outils actifs de l'atelier-logiciel IDA.
"Le logiciel DSL-PROTO est un outil de création d'un maquette fonctionnelle d'un système d'information qui permet au concepteur de ce système d'information, aux analystes et aux utilisateurs qui ont participé à la conception de ce système d'information de déterminer expérimentalement si les spécifications produisent les résultats qu'ils attendent.
Une maquette est une réalisation à échelle réduite du système d'information projeté.
DSL-PROTO génère des maquettes par la transformation partiellement automatique des spécifications relatives aux informations et aux traitements en instructions exécutables." [PRO-86]

8.2.2. ARCHITECTURE DE DSL-PROTO

L'architecture de DSL-PROTO est représentée à la figure 8.1.
On y trouve six modules.
Pour établir un parallélisme avec la présentation du logiciel ORACLE faite au chapitre 7, nous classifions ces différents modules en deux catégories représentant chacune une étape de développement d'un prototype avec DSL-PROTO (même si pour ORACLE, on pouvait parler plus précisémment d'outils de développement).
Les deux étapes sont :
- la génération de la maquette (comprenant les modules MDYN, MDTS, MSTT, MTRN, MUSM) (point 8.2.3.)
- l'exécution de la maquette (comprenant le module MEXE) (point 8.2.4.).

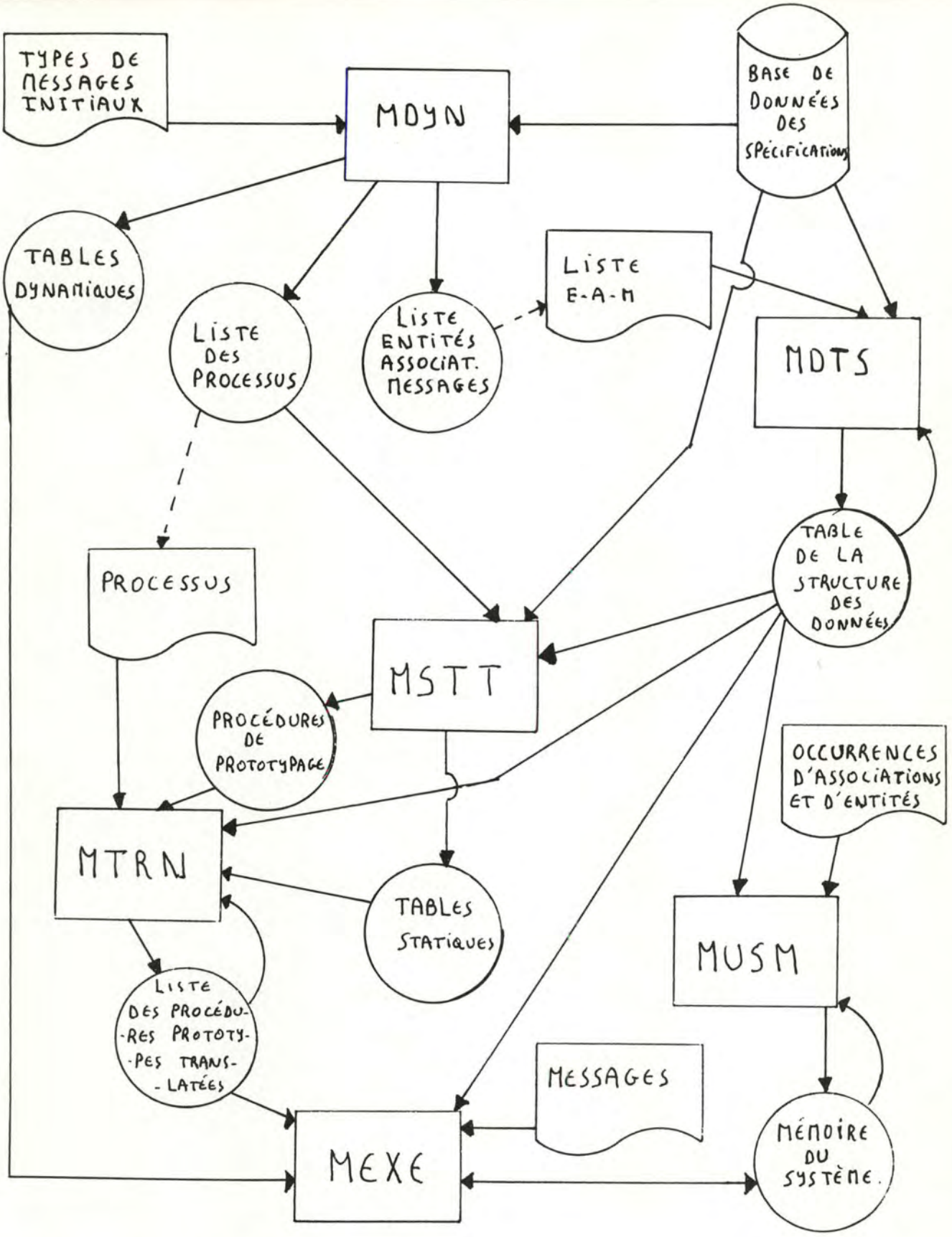


figure 8.1. Architecture de DSL-PROTO (source : [PRO-86])

8.2.3. GENERATION DE LA MAQUETTE

La génération de la maquette, qui est la première étape de développement d'un prototype DSL-PROTO, est composée de sept étapes, que nous énumérons ci-dessous.

ETAPE 1 : REMPLISSAGE DE LA BASE DE DONNEES DES SPECIFICATIONS
ETAPE 2 : GENERATION DE L'ENVIRONNEMENT STATIQUE
ETAPE 3 : CREATION DES TABLES DE DEFINITION DE LA STRUCTURE DE DONNEES
ETAPE 4 : GENERATION DE L'ENVIRONNEMENT STATIQUE
ETAPE 5 : EXTRACTION DES PROCEDURES DE PROTOTYPAGE
ETAPE 6 : TRADUCTION DES PROCEDURES DE PROTOTYPAGE
ETAPE 7 : MISE A JOUR DE LA MEMOIRE DU SYSTEME

8.2.4. EXECUTION DE LA MAQUETTE

L'exécution de la maquette, qui est la seconde étape de développement d'un prototype DSL-PROTO, est réalisée par le module MEXE.
L'utilisateur fournit à ce module des occurrences de messages externes à partir desquels il désire que l'exécution se déroule.
"A partir de ces occurrences de messages externes, les tables de la dynamique déterminent les processus à exécuter, leur condition de déclenchement et leur synchronisation.
Le code des procédures correspondant à ces traitements est fourni par les tables du pseudo-langage ; les types d'objets manipulés par ces traitements sont décrits dans les tables de définition de la structure de données ; les valeurs initiales des types d'entités et d'associations ont été enregistrées dans la mémoire du système.
En cours d'exécution, les procédures mettront à jour la mémoire du système." [PRO-86]
Il y a deux modes d'exécution :
"- un mode réexécutable qui permet de réexécuter plusieurs fois un même traitement (transaction); dans ce mode, il n'y a pas d'enregistrement des changements d'état (ajout, suppression ou modification) de la mémoire du système qui résultent de l'exécution d'une transaction (par transaction, on entend l'exécution d'un prototype associée à la réception d'un message externe)
- un mode cumulatif où chaque changement d'état intervenant lors d'une transaction est enregistré dans la mémoire du système.
Au niveau des deux modes d'exécution, le module MEXE demande à l'utilisateur le niveau de trace qu'il désire." [PRO-86]

8.3. FONCTIONS A REALISER

Les fonctions à réaliser sont celles qui proviennent de la phase "Répartition-des-engagements" de l'application "Plan de pièces".

Pour rappel, nous énumérons ces fonctions, dont la dynamique et la statique ont été décrites aux points 5.4.1. et 5.4.2.

Fonction 1 : sélection-lignes-actives
Fonction 2 : sélection-lignes-inactives
Fonction 3 : traduction-cadence
Fonction 4 : choix-lignes-possibles
Fonction 5 : création-nouvelles-lignes
Fonction 6 : mise-à-jour-engagements
Fonction 7 : création-engagements
Fonction 8 : contrôle-validité-engagement
Fonction 9 : contrôle-total-engagement
Fonction 10 : validation-plan-pièces

Le sous-schéma conceptuel des données de la phase "Répartition-des-engagements" a été présenté à la figure 7.2.

8.4. LES TRANSFORMATIONS EFFECTUEES

Nous avons vu précédemment la structuration des données, la structuration, la dynamique et la statique des traitements des fonctions à réaliser.
Nous montrons dans ce point les transformations effectuées pour réaliser ces fonctions avec le logiciel DSL-PROTO.
Il y a deux types de transformations :
- les transformations volontaires, que nous avons effectuées de plein gré, et qui ne sont pas imposées par les fonctionnalités du logiciel DSL-PROTO
- les transformations nécessaires, autrement dit les adaptations par rapport à la dynamique et la statique des traitements découlant de l'analyse fonctionnelle.

Il est clair qu'il n'y a pas de changement à apporter à la structuration des données.

A titre indicatif, nous donnons tout d'abord le schéma de la dynamique résultant de l'analyse fonctionnelle (figure 8.2.) (même schéma qu'à la figure 5.2.) et le schéma de la dynamique transformé (figure 8.3.).
Puis le code DSL de la dynamique est donné.

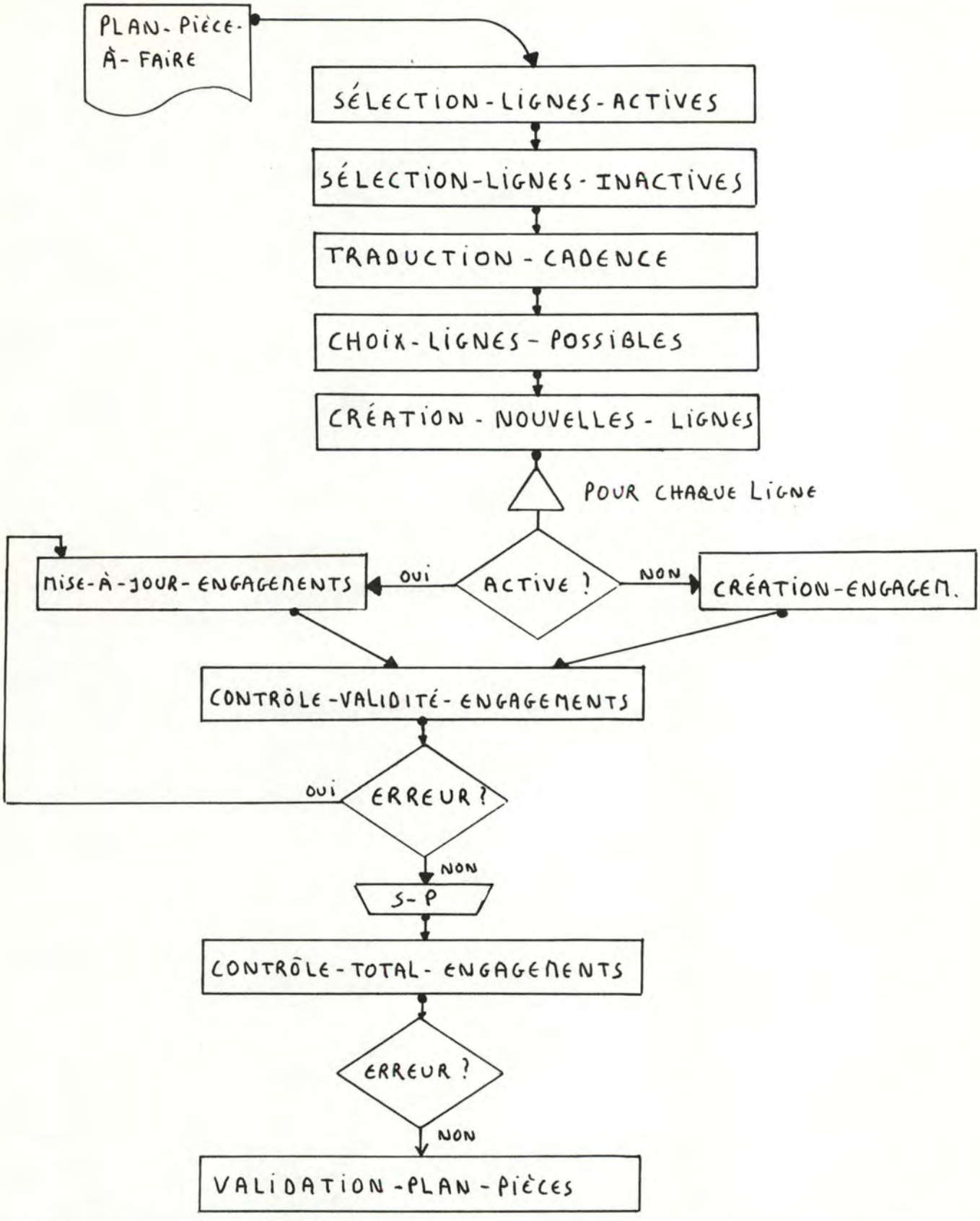


figure 8.2. Dynamique de la phase "Répartition-des-engagements"

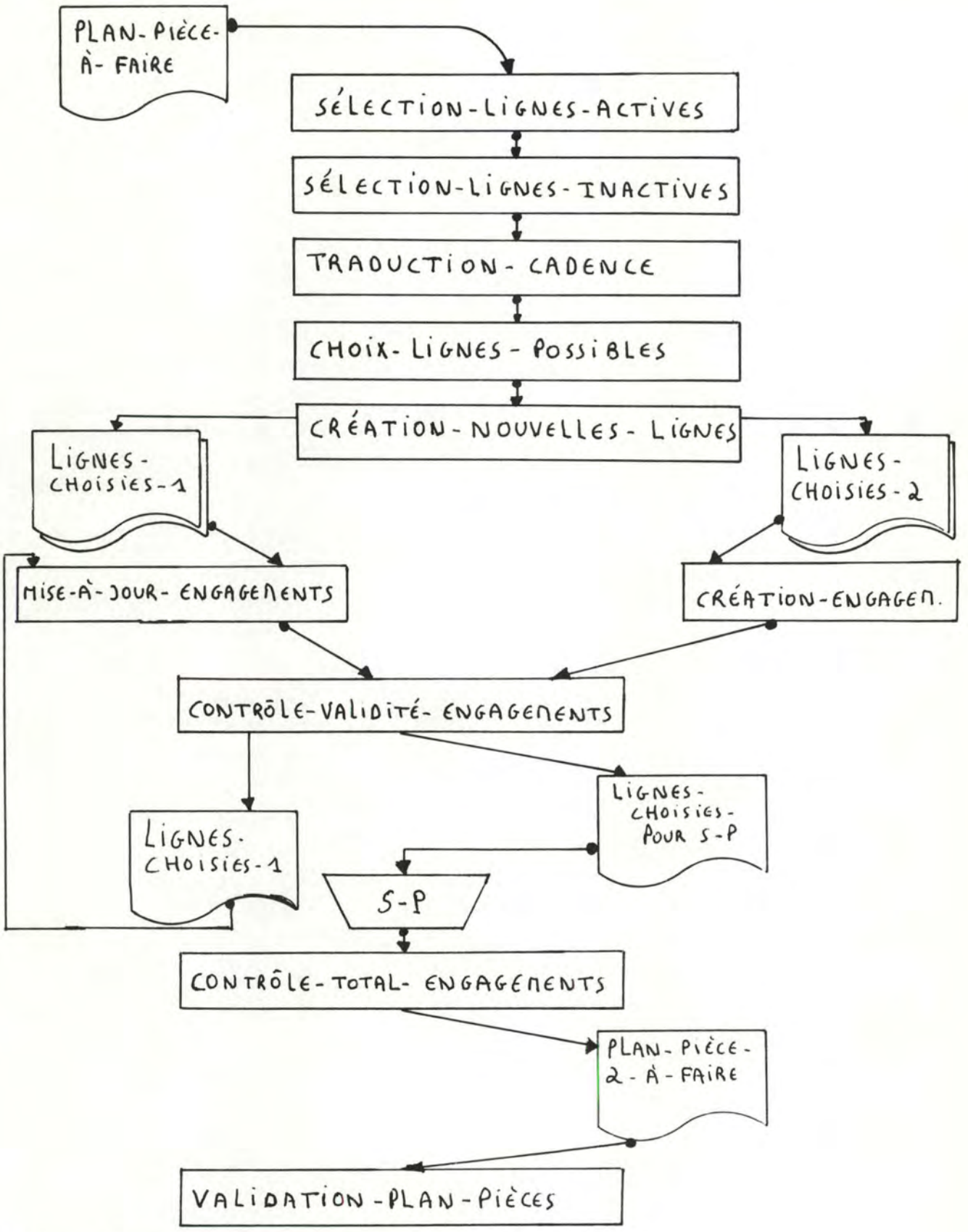


figure 8.3. dynamique transformée

Le code DSL de la dynamique est :

1) sélection-lignes-actives :

TRIGGERED BY GENERATION OF plan-piece-a-faire

2) sélection-lignes-inactives :

TRIGGERED BY TERMINATION OF sélection-lignes-actives

3) traduction-cadence :

TRIGGERED BY TERMINATION OF sélection-lignes-inactives

4) choix-lignes-possibles :

TRIGGERED BY TERMINATION OF traduction-cadence

5) création-nouvelles-lignes :

TRIGGERED BY TERMINATION OF choix-lignes-possibles

6) mise-à-jour-engagements :

TRIGGERED BY GENERATION OF lignes-choisies-1

7) création-engagements :

TRIGGERED BY GENERATION OF lignes-choisies-2

8) contrôle-validité-engagement :

TRIGGERED BY TERMINATION OF mise-à-jour-engagements
TRIGGERED BY TERMINATION OF création-engagements

9) contrôle-total-engagement :

TRIGGERED BY REALIZATION OF pt-synchro

10) validation-plan-pièces :

TRIGGERED BY GENERATION OF plan-piece-2-a-faire

**) pt-synchro :

CONTRIBUTED BY GENERATION OF lignes-choisies-pour-p-s
REALIZED-WHEN ;
GROUP-ALL nbre-message TERM OF contrôle-validité-engagement

8.4.1. TRANSFORMATIONS VOLONTAIRES

Il y a deux transformations volontaires : la première concerne la dynamique par les processus et/ou les messages et la seconde concerne les conditions.

A) Dynamique par les processus et/ou les messages

Le schéma initial décrit la dynamique par les processus.
Néanmoins, avec DSL-PROTO, il est possible de décrire la dynamique soit par les processus, soit par les messages, les deux techniques n'étant pas exclusives.
Le schéma transformé (figure 8.3.) présente les deux formes.

Il est clair qu'il ne s'agit pas d'une transformation due aux fonctionnalités de DSL-PROTO. Simplement, nous n'avons pas pris en considération, lors de l'analyse fonctionnelle dont découle le schéma initial de la dynamique, la possibilité de dynamique par les messages.

B) Les conditions

Par rapport au schéma initial, les conditions de déclenchement sont supprimées.

- La première condition (après la fonction "création-nouvelles-lignes") est supprimée car les fonctions "mise-à-jour-engagements" et "création-engagements" sont déclenchées par la génération de messages de **noms différents**.

- La deuxième condition (après la fonction "contrôle-validité-engagement") est supprimée car la contribution au point de synchronisation est établie par la génération d'un message qui n'est généré par le processus "contrôle-validité-engagements" que s'il n'y a pas d'erreur, en d'autres mots que si la contribution au point de synchronisation peut avoir lieu.

- La troisième condition (après la fonction "contrôle-total-engagement") est supprimée pour les mêmes raisons que la deuxième.

Ces transformations sont volontaires, vu qu'elles découlent de la construction de la dynamique par les messages.

8.4.2. TRANSFORMATIONS NECESSAIRES

Les seules transformations nécessaires concernent la réalisation du déclenchement multiple.

Le schéma initial contient un "FOR EACH lignes choisies ou créées" après la fonction "création-nouvelles-lignes". Tous les processus entre le FOR EACH et le point de synchronisation sont à exécuter pour chaque ligne choisie ou créée.
Mais DSL-PROTO, dans le cas d'une construction "FOR EACH attribut", ne prend en compte qu'une description par les messages, ceci pour des raisons d'implémentation.
Par conséquent, la fonction "mise-à-jour-engagements" est déclenchée chaque fois qu'un message "lignes-choisies-1" est généré, et la fonction "création-engagements" est déclenchée chaque fois qu'un message "lignes-choises-2" est généré.

Remarque : le point de synchronisation est réalisé par la clause

GROUP-ALL nbre-message TERM OF contrôle-validité-engagement

et "nbre-message" est un attribut dont la valeur est donnée par la clause

DYNAMICS VALUE IS CARDINALITY OF compteur-lignes
ON TERMINATION OF création-nouvelles-lignes.

Or, "CARDINALITY OF" ne peut porter que sur **un seul** message. Par conséquent, la fonction "création-nouvelles-lignes" doit générer les occurrences du message compteur-lignes. Celles-ci correspondent aux occurrences de "lignes-choisies-1" et aux occurrences de "lignes-choisies-2". Il y a donc duplication des messages générés.

8.5. IMPLEMENTATION

Nous montrons dans ce point l'implémentation d'une fonction avec DSL-PROTO (fonction 1). L'annexe 4 contient l'implémentation de toutes les fonctions, ainsi qu'une implémentation des aides à l'élaboration des hypothèses.

Fonction 1 : sélection-lignes-actives

```
FOR EACH ligne-product
    FOR EACH montage-boite-v
        WHERE nom-ligne OF montage-boite-v == nom-ligne OF
              ligne-product
        WITH COUNTER compteur
        ** on a trouvé une ligne active.
        [ Calcul du total de ses engagements pour chaque
        période. ]
    ENDFOR montage-boite-v
    FOR EACH montage-chassis... ¦
    FOR EACH montage-moteur.... ¦ même traitement que pour boite-v
    FOR EACH usinage.......... ¦
    IF compteur == 0
    THEN
        ** la ligne est inactive et on passe à la ligne suivante
        NEXT BLOCK-FOR ligne-product
    ELSE
        [ Vérification que, pour chaque période, les engagements
          existants sont inférieurs aux capacités. ]
        IF oK
        THEN
            ** on peut sélectionner la ligne
            [ Génération d'un message contenant le nom de la
              ligne. ]
        ENDIF
    ENDIF
ENDFOR ligne-product
```

Commentaires :
Le principe consiste à, pour chaque occurrence de ligne de production, et pour chaque occurrence de montage ou d'usinage qu'elle possède, sommer les engagements. Un compteur (instruction WITH COUNTER) permet de connaître le nombre d'occurrences de montage ou d'usinage trouvées. Si ce compteur vaut 0 en sortie des boucles portant sur le montage ou l'usinage, la ligne est inactive et on passe à l'occurrence suivante de ligne-product. Sinon, on vérifie que la ligne est encore capable de monter ou d'usiner (c'est-à-dire que ses engagements sont inférieurs à sa capacité).

8.6. ADAPTATIONS ET EVOLUTIONS

Nous avons effectué certaines adaptations, dues non pas aux fonctionnalités de DSL-PROTO, mais aux limites temporaires de l'outil utilisé en cours de tests.
La prochaine version du logiciel DSL-PROTO, disponible à partir du mois de septembre 86, contient les remèdes aux limites décrites ci-dessous.

8.6.1. LIMITES DUES A L'OUTIL

Il y a trois limites dues à l'outil dans sa version employée lors des tests : l'absence de variable indicée de travail, l'absence de "move corresponding" et l'absence de qualification elliptique.

A) Absence de variable indicée de travail

Elle peut être contournée en substituant aux variables indicées de travail, c'est-à-dire locales à une procédure, des messages "de travail".
Ceci a la conséquence suivante : étant donné que pour chaque fonction implémentée avec DSL-PROTO, il doit exister une correspondance précise entre toutes les phrases DSL de la statique et les primitives du langage d'expression d'algorithmes et que nous avons pris l'option de substituer aux variables indicées de travail des messages "de travail", il faut, afin de pouvoir accéder au contenu de ces messages, que ceux-ci fassent partie d'un bloc de réception de message, et vu le principe de correspondance, que les messages de travail "reçus" soient énumérés dans la statique des fonctions.

Ceci entraîne une **adaptation de la statique des traitements.**

B) Absence de "move corresponding"

Lorsque le concepteur veut recopier tous les éléments d'une entité, d'une association ou d'un message dans un autre type d'objet de même nature, il est obligé d'écrire autant d'instructions d'assignation qu'il y a de zones à recopier.
L'instruction d'assignation globale (équivalente à un "move corresponding" du langage COBOL) simplifiera l'écriture.

C) Absence de qualification elliptique

Un même nom d'élément pouvant appartenir à plusieurs entités, associations ou messages, il est nécessaire de qualifier chaque groupe ou élément par le type d'objet dont ils dépendent.
Exemple de forme possible : code-postal OF adresse OF client.
Cette écriture peut être longue et fastidieuse, surtout si l'élément et/ou le groupe est indicé.
La qualification elliptique qui sera ultérieurement implémentée soulagera l'écriture.
On pourra dès lors avoir : code-postal OF client

8.6.2. PROBLEMES METHODOLOGIQUES

Nous voulons traiter dans ce point des problèmes méthodologiques rencontrés.
Le premier problème présenté (qui est minime) concerne le principe de localité et a des conséquences sur la description de la statique des traitements.
Le deuxième problème a pour objet la structuration des traitements en fonction de l'introduction des dialogues.
Le troisième problème, important, concerne la gestion des menus.

A. LE PRINCIPE DE LOCALITE

"Un processus ne peut recevoir des messages que du ou des seuls événements qui l'ont déclenché et/ou de l'environnement." [PRO-86]. Il s'agit du **principe de localité**.
Par conséquent, dans notre dynamique, si une fonction (i+1) a besoin du contenu d'un message généré par une fonction (i-1), la fonction (i) doit recevoir et générer ce message à destination de la fonction (i+1). Le lecteur ne s'étonnera donc pas de trouver dans la statique associée au code d'une fonction (cfr annexe 6) des réceptions et des générations de messages "en transit".

Ceci entraîne une adaptation de la statique des traitements par rapport à celle résultant de l'analyse fonctionnelle.

B. MODIFICATIONS DUES AU DIALOGUE

Le dialogue entre l'utilisateur et l'application modifie quelque peu les objectifs assignés aux fonctions lors de l'analyse fonctionnelle.
Rappelons d'abord que l'application dialogue avec l'utilisateur, au point de vue technique, soit par la génération de messages à destination de l'environnement, soit par l'affichage à l'écran pur et simple (fonction DISPLAY). L'utilisateur quant à lui dialogue par messages reçus par l'application.
Le principe du dialogue par génération-réception de messages implique qu'il y ait rupture dans la découpe en fonctions entre la question posée par l'application et la réponse apportée par l'utilisateur.

Comme illustration, reprenons la spécification de l'objectif de la fonction 4 ("choix-lignes-possibles") : "permettre à l'utilisateur de choisir, parmi les lignes actives et/ou inactives proposées, celles sur lesquelles ... Vérification que les lignes données par l'utilisateur existent dans les lignes actives et inactives."

Le dialogue est possible si la fonction 3 génère un message contenant les lignes proposées ainsi qu'un affichage à l'écran avertissant l'utilisateur qu'il doit choisir certaines lignes et si la fonction 4 reçoit un message contenant les lignes choisies par l'utilisateur.

CONSEQUENCE :

une conséquence de l'implémentation des dialogues est l'ajout d'un nouveau critère de découpe en fonction, par rapport à ceux développés dans [BOD-83], à savoir qu'il ne peut y avoir aucun dialogue à l'intérieur d'une fonction. Ceci entraîne que la découpe d'une phase en fonctions est particulièrement fine.

C. LA GESTION DES MENUS

Il est clair que dans la dynamique des fonctions de la phase "Répartition-des-engagements" que nous avons choisi de développer, la démarche est **séquentielle**.

Si elle est séquentielle, c'est parce que nous avons fait le **choix** de l'établir de la sorte.

C'est une **restriction**.

Si cette restriction peut encore se comprendre lorsqu'on a pour but d'utiliser DSL-PROTO comme outil de vérification des spécifications à destination principale du concepteur de l'application, elle est nettement moins justifiable et compréhensible lorsqu'on désire assigner à DSL-PROTO un but d'apprentissage, d'expérimentation par l'utilisateur final, voire de décision d'investissements par le décideur financier.

Ce qui est le cas dans le prototypage de l'application "plans de pièces", ainsi que nous l'avons déjà dit.

Cette séquentialité est d'autant plus restrictive que le logiciel ORACLE, par la gestion des menus qu'il offre (sous la forme de menus affichés à l'écran ou par la technique des touches-fonctions) permet à l'utilisateur de voyager à son gré dans les différents blocs et écrans de l'application.

La seule justification à la dynamique séquentielle que nous avons implémentée est que nous avons choisi délibéremment de le faire pour des raisons de simplification. C'est donc une démarche de travail tout à fait **volontaire** et qui n'est pas imposée par les fonctionnalités de l'outil.

Il est alors intéressant, à ce stade, de montrer jusqu'où nous aurions pu pousser l'outil DSL-PROTO, c'est-à-dire de permettre de voir comment et avec quelles méthodes on aurait pu présenter une dynamique effectivement basée sur les choix de l'utilisateur. Nous avons choisi d'avoir une démarche en gradation, qu'on pourrait appeler "par essais-erreurs".

On peut d'abord concevoir une dynamique non plus purement séquentielle, mais possédant un enchaînement éclaté et un enchaînement convergent.

La fonction 3 (traduction-cadence) peut être effectuée avant les fonctions 1 et 2 (sélection des lignes actives et inactives). La fonction 4 doit être effectuée après les fonctions 1, 2, 3.
Cette séquence est schématisée à la figure 8.4.

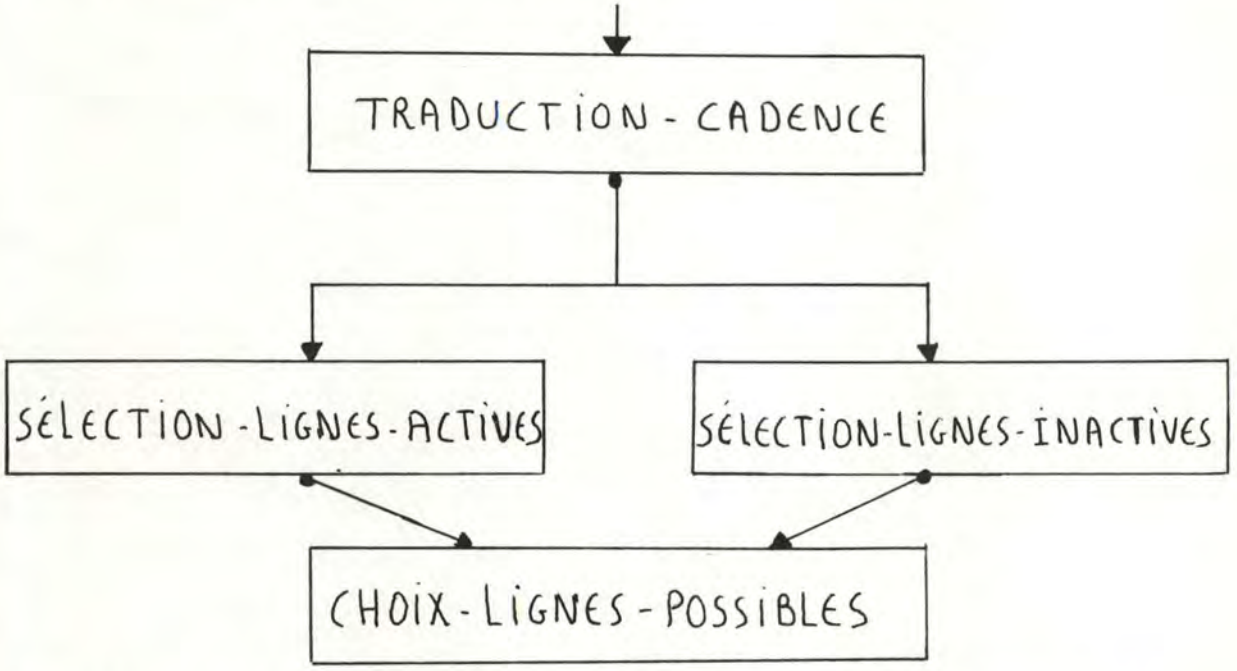


figure 8.4. dynamique avec enchaînements éclaté et convergent

La terminaison de "traduction-cadence" provoque le déclenchement simultané de "sélection-lignes-actives" et "sélection-lignes-inactives".
Ce type d'enchaînement permet de ne pas imposer de séquence au niveau de la sélection des lignes. Néanmoins, on remarque que l'utilisateur n'a pas encore le choix de sélectionner d'abord les lignes actives ou d'abord les lignes inactives.

D'autre part, le déclenchement de "choix-lignes-possibles" est provoqué par la terminaison de "sélection-lignes-actives" **OU** de "sélection-lignes-inactives".
On pourrait donc avoir, au niveau de l'exécution, la séquence "traduction-cadence", "sélection-lignes-actives", "choix-lignes-possibles", "sélection-lignes-inactives", "choix-lignes-possibles".

Ceci ne nous satisfait pas, puisque le choix, d'après les spécifications, doit se faire sur base des lignes actives et inactives proposées.

Ceci est réalisé en remplacant l'enchaînement convergent par un enchaînement synchronisé. (figure 8.5.)

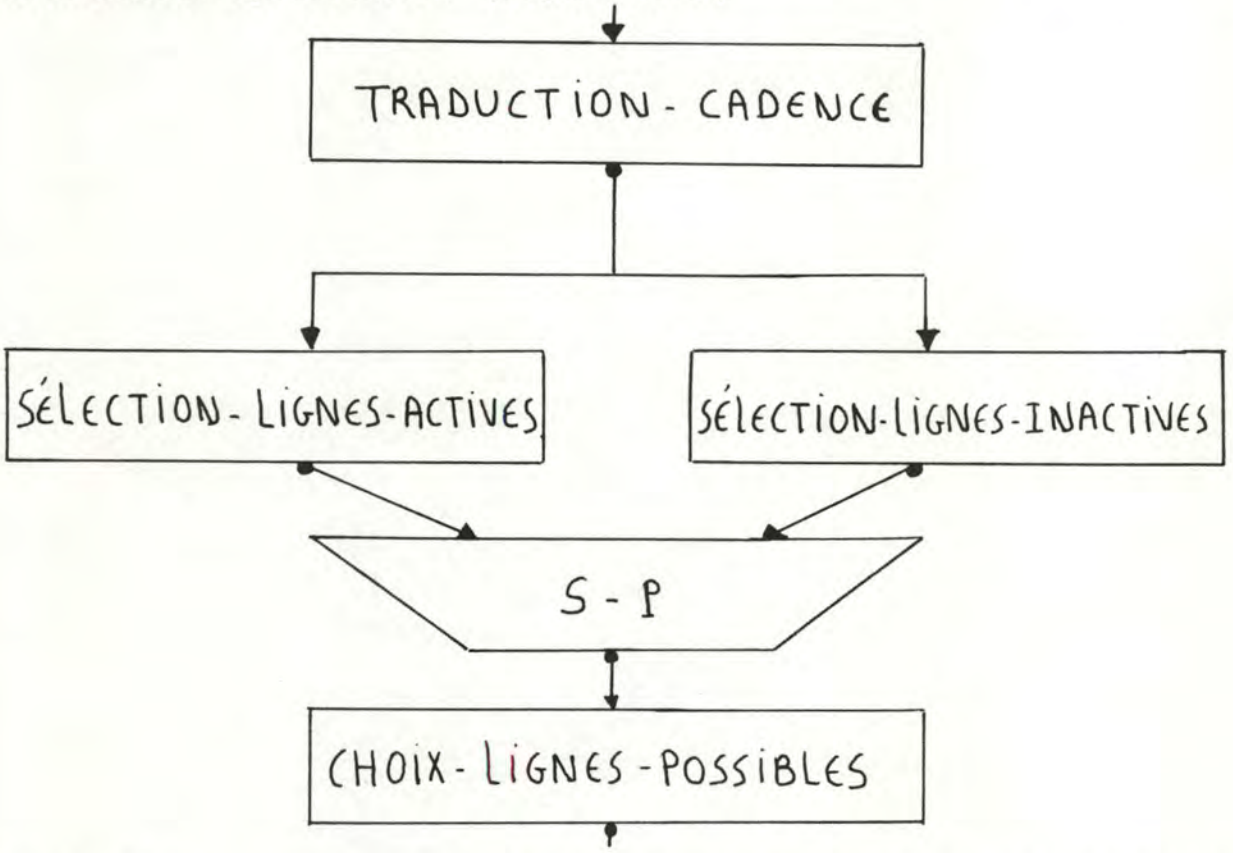


figure 8.5. dynamique avec introduction d'un enchaînement synchronisé

Dans ce cas, le déclenchement du processus "choix-lignes-possibles" dépend de la terminaison des processus "sélection-lignes-actives" et "sélection-lignes-inactives".
La condition de synchronisation est 'terminaison de "sélection-lignes-actives" et terminaison de "sélection-lignes-inactives"'.

Cette solution, améliorée, ne nous satisfait pas encore entièrement, dans la mesure où l'utilisateur ne peut pas encore choisir.
On peut alors introduire des fonctions spéciales de choix, dont le but de demander à l'utilisateur le chemin qu'il désire parcourir.

Soit l'introduction d'une fonction "choix-chemin" déclenchée par la terminaison de "traduction-cadence" et recevant un message "choix" contenant une variable indiquant le choix de chemin de l'utilisateur (il s'agit de la technique de dialogue développée au point B. ci-dessus).
On introduit aussi un enchaînement conditionnel.

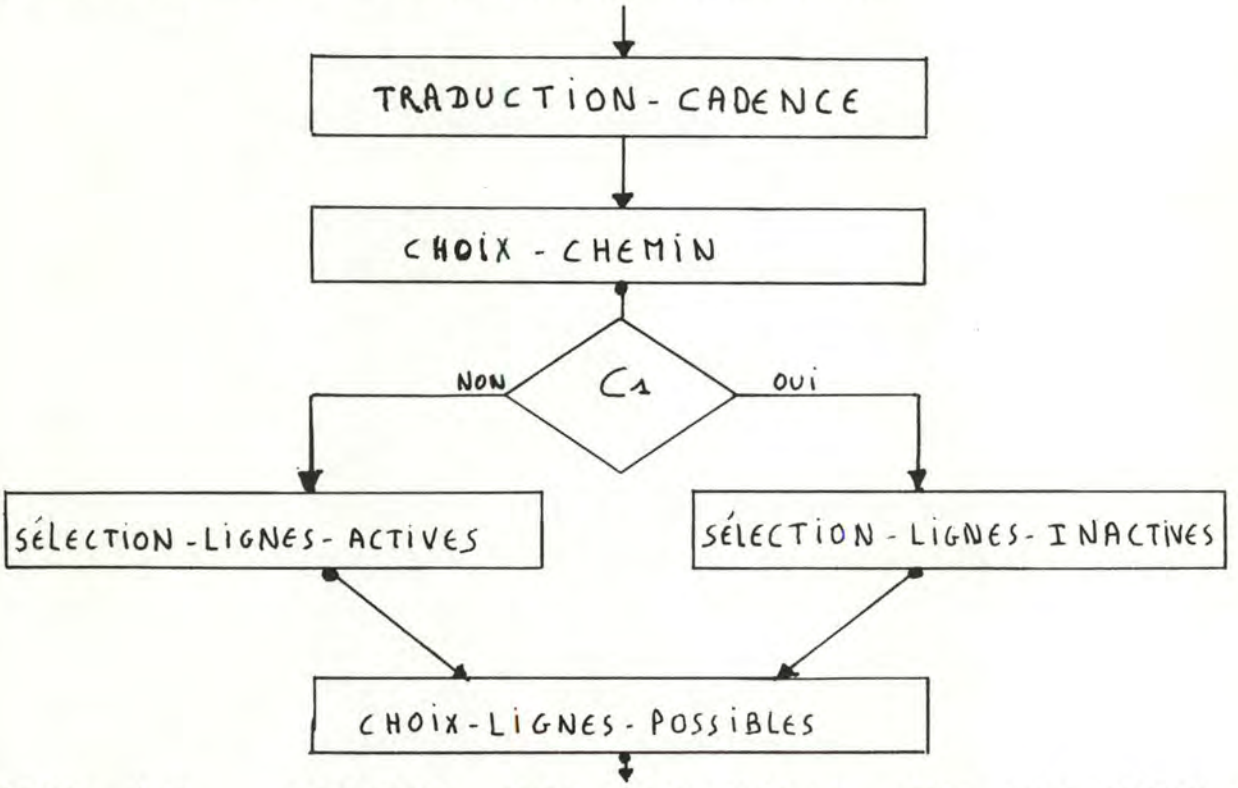


figure 8.6. dynamique avec introduction d'un enchaînement conditionnel

La condition C1 est vraie si l'utilisateur a choisi de sélectionner les lignes inactives, fausse s'il désire sélectionner les lignes actives.
La terminaison de "choix-chemin" déclenche "sélection-lignes-inactives" si C1 est vraie, "sélection-lignes-actives" sinon.

Cette solution est imparfaite dans la mesure où, dans ce cas, on ne passe plus que par **un** des deux chemins.
On peut contourner ce problème en ajoutant encore une condition en terminaison de "sélection-lignes-actives" et en terminaison de "sélection-lignes-inactives".

La combinaison de toutes ces techniques permet de répondre à tous les choix de l'utilisateur.
On peut donc implémenter artificiellement la notion de menu.

La meilleure solution, qui n'est pas encore implémentée, est l'introduction de menus, qui permettent à l'utilisateur de déterminer le chemin qu'il désire effectuer.

L'utilisateur pourrait dans ce cas répondre qu'il désire le déclenchement de la fonction 1, puis de la fonction 2, ou le contraire, ou éventuellement le déclenchement de la fonction 1 (ou 2) uniquement, ou qu'il ne désire aucun déclenchement (ce qui est envisageable sémantiquement en règle générale, mais pas dans notre cas). On pourrait imaginer de revenir au menu tant qu'on ne décide pas d'avancer plus loin (c'est-à-dire passer au menu suivant) et même de voyager (en avant ou en arrière) de menu en menu, pourvu que la sémantique et les spécifications des fonctions le permettent.
On obtient alors le schéma de la figure 8.7. où l'introduction d'un menu est schématisée par un triangle renversé contenant un point d'interrogation.

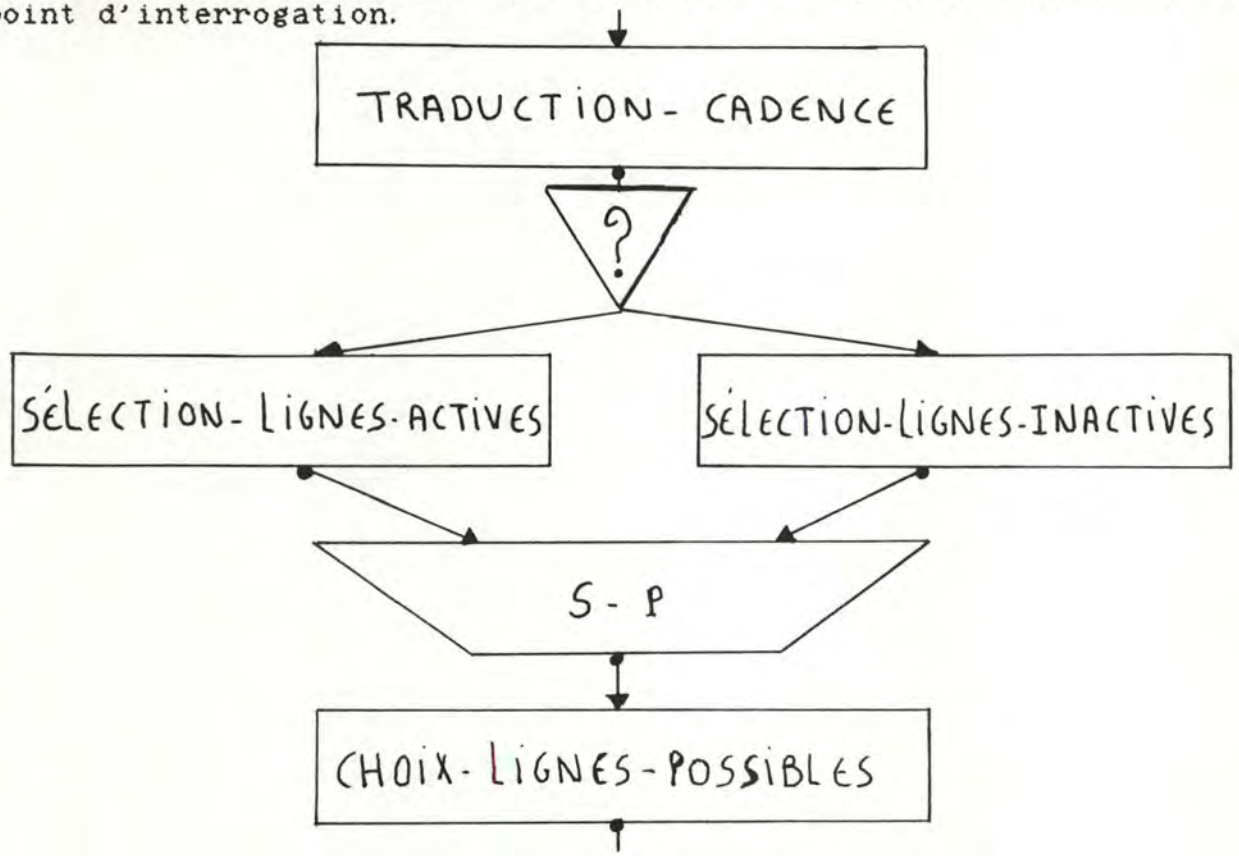


figure 8.7. dynamique avec introduction d'un menu

CONCLUSION

L'introduction de menus permet à l'utilisateur de faire des choix dans l'ordre d'exécution des différents processus.

Ceci est très important lorsque, comme c'est le cas dans le prototypage de l'application "plans de pièces", le prototypage est notamment destiné à une utilisation finale.
La séquentialité implémentée est donc une restriction volontaire.

La démarche en gradation que nous venons d'effectuer a permis de voir jusqu'à quelles limites on pouvait pousser l'outil DSL-PROTO, et de constater que moyennant l'introduction de menus, on pouvait permettre facilement à l'utilisateur de "gérer sa dynamique".

CHAPITRE 9 : COMPARAISON

INTRODUCTION

Le but de ce neuvième et dernier chapitre est d'évaluer et de comparer les deux outils, DSL-PROTO et ORACLE, dans le cadre du prototypage de l'application "plans de pièces".

Nous commençons par développer les critères de comparaison. Ensuite, nous évaluons les deux logiciels critère par critère.

Les critères peuvent être répartis en quatre grandes catégories.

La première catégorie concerne les critères de fonctionnalités du prototypage.

La deuxième catégorie concerne les critères de fiabilité du prototypage.

La troisième catégorie concerne les critères de développement du prototype.

La quatrième catégorie concerne les critères de l'exploitation du prototype.

Des deux logiciels, DSL-PROTO et ORACLE, nous n'avons pas voulu, en général, préférer critère par critère un logiciel à l'autre. Nous n'avons pas voulu non plus tirer une conclusion globale de prédominance d'un logiciel sur l'autre, ne fût-ce que dans le cas de l'application "plans de pièces".

En effet, cette conclusion résulterait de la sommation de résultats attribués à chaque critère, ce qui est impossible tant il est ardu et subjectif de pondérer dans la somme l'importance à donner non seulement à chaque catégorie de critères, mais aussi à chaque critère au sein d'une même catégorie.

9.1. LES CRITERES DE COMPARAISON

Nous présentons dans ce point les critères de comparaison. Ils se répartissent en quatre catégories.

9.1.1. LES CRITERES DE FONCTIONNALITES

Les critères de fonctionnalités du prototypage sont extrêmememt nombreux. Aussi nous sommes-nous limités à quelques-

uns d'entre eux, à savoir la consultation de la base de données, la mise-à-jour de la base de données, la création de l'application, l'écriture d'algorithmes, l'exécution.
Nous détaillons ci-dessous ces critères.

A. CONSULTATION DE LA BASE DE DONNEES

Dans quelle mesure peut-on consulter la base de données? L'accès est-il facile? Il est intéressant de voir aussi quelles données on peut consulter. Nous verrons dans quelle mesure on peut consulter aussi les spécifications de la dynamique et de la statique des traitements, les spécifications des messages, etc...

B. MISE-A-JOUR DE LA BASE DE DONNEES

Nous verrons dans quelle mesure on peut mettre à jour la base de données des spécifications, dans le même ordre d'idées qu'au point A. développé ci-dessus.

C. CREATION DE L'APPLICATION

Par rapport aux spécifications fonctionnelles, nous verrons si on peut facilement créer une application, et sous quelles conditions.

D. ECRITURE D'ALGORITHMES

Nous pensons qu'il est intéressant de voir comment on peut transcrire l'algorithme manipulant les données d'une fonction pour réaliser l'objectif assigné à cette fonction.

E. EXECUTION

Nous désirons comparer les deux logiciels quant aux fonctionnalités qu'ils offrent au point de vue de l'exécution du prototype. Peut-on exécuter l'application plusieurs fois sur les mêmes données, peut-on modifier l'état de la base de données lors de l'exécution? Peut-on reprendre une exécution à un certain niveau d'avancement? Dans quelle mesure l'utilisateur peut-il apprendre le fonctionnement du futur système.

9.1.2. LES CRITERES DE FIABILITE

Il y a deux critères de fiabilité du prototypage : le respect des spécifications et le contrôle du respect des spécifications.

A. LE RESPECT DES SPECIFICATIONS

Le respect des spécifications est un critère important de la fiabilité du prototypage.
Respecte-t'on, lors du prototypage de l'application, les spécifications, c'est-à-dire :
- au point de vue des données : la structuration des données telles que décrites dans le schéma conceptuel des données (cfr chapitre 3)
- au point de vue des traitements : la structuration, la statique et la dynamique des traitements (cfr chapitre 5).

B. LE CONTROLE DU RESPECT DES SPECIFICATIONS

Dès lors qu'on considère qu'un critère est le respect des spécifications, il est clair qu'il est intéressant de voir dans quelle mesure chaque logiciel permet de contrôler si les spécifications sont respectées.

9.1.3. LES CRITERES DE DEVELOPPEMENT

On peut considérer qu'il y a trois critères concernant le développement du prototype : la simplicité de réalisation, l'évolutivité et la modularité.

A. LA SIMPLICITE DE REALISATION

Nous envisageons la simplicité de réalisation à deux niveaux : la forme et le contenu de la réalisation :
* la forme de la réalisation :
- le programme est-il long à écrire? Y-a-t'il des répétitions,...?
- le programme est-il difficile à écrire en termes de réalisation physique (éditeur, dialogue, ...)?
* le contenu de la réalisation :
- à partir des spécifications, le prototype est-il facile à mettre en oeuvre?
- le programme est-il difficile à écrire en termes de langage?

Tout ce critère découle, dans une certaine mesure, des fonctionnalités offertes, décrites ci-dessous.

Ce critère correspond à **l'ergonomie du concepteur**.

B. EVOLUTIVITE

Ainsi que nous l'avons souligné au point 6.3.2., l'évolutivité est un critère de comparaison important, dans la mesure où elle découle de la typologie de prototypage souhaitée par les utilisateurs finals (prototypage évoluant de version en version)

et dans la mesure où pour le concepteur, le prototypage est un moyen d'acquisition et de consolidation de connaissances et de spécifications du problème à résoudre.
Cette évolutivité peut être envisagée à deux niveaux : les structures de données et les traitements. A ces deux niveaux, les modifications sont-elles faciles à réaliser?

C. MODULARITE

Au point de vue du concepteur, le prototypage de l'application "plans de pièces" doit pouvoir être intégré à un prototype plus vaste, celui du projet "Filière Méthodes Mécaniques".
Dans ce but, il est important que le prototypage offre une modularité.

9.1.4. LES CRITERES D'EXPLOITATION

Nous considérons qu'il y a quatre critères à prendre en compte dans l'évaluation de l'exploitation du prototype : la facilité d'exploitation, l'ergonomie, le temps de réponse et la confidentialité.
Ces critères sont importants dans un but d'utilisation du prototype non seulement en tant qu'outil d'apprentissage, mais aussi de démonstration pour la décision d'investissement.

A. FACILITE D'EXPLOITATION

L'outil doit être facile à exploiter pour diverses raisons.
Outre celles présentées ci-dessus, à savoir l'utilisation comme outil d'apprentissage et comme argument de décision, il ne faut pas négliger que les utilisateurs ont des compétences réduites en informatique, qu'ils ont une faible propension à apprendre. (cfr point 6.3.1.)

B. ERGONOMIE

L'ergonomie est capitale, pour les raisons déjà expliquées.
Nous soulignons en plus que dans la mesure où le prototype de l'application "plans de pièces" est considéré comme la version intermédiaire d'un produit fini, ce prototype préfigure aux utilisateurs ce que seront leurs conditions de travail.

C. TEMPS DE REPONSE

Le temps de réponse est un critère intéressant. En effet, il ne faut pas que l'utilisateur, qu'il s'agisse de l'utilisateur final futur ou du décideur, ait l'impression de perdre son temps

pendant des recherches trop longues d'informations dans la base de données, surtout qu'il est prouvé par expérimentation que l'énervement produit par trop d'attentes inactives conduit à l'exécution de fausses manoeuvres involontaires.
Le temps de réponse peut même être considéré comme un élément d'ergonomie pour l'utilisateur.

D. CONFIDENTIALITE

Comme nous l'avons expliqué au point 6.3.5., la confidentialité est très importante pour les utilisateurs.
Or, il est normal que dans le cadre d'un prototypage destiné à l'utilisation, on travaille sur des objets existants, afin de s'y retrouver au niveau des données et afin de se familiariser mieux et plus au nouveau produit.
Ceci est d'autant plus vrai qu'on désire intégrer le prototypage de l'application "plans de pièces" dans un contexte plus large, qui dépasse les frontières de l'espace privé des utilisateurs du prototype.
La confidentialité peut aussi se concevoir au niveau des traitements.

9.2. EVALUATION ET COMPARAISON

Nous reprenons les quatre catégories de critères de comparaison développées au point 9.1. et nous appliquons ces critères à l'évaluation et la comparaison de DSL-PROTO et ORACLE.

9.2.1. LES FONCTIONNALITES DU PROTOTYPAGE

Les cinq critères ont été détaillés au point 9.1.1.

A. CONSULTATION DE LA BASE DE DONNEES

Selon des modalités différentes, ORACLE et DSL-PROTO permettent la consultation de la base de données. Cette fonctionnalité, globalement, est équivalente pour les deux logiciels. Seules les formes de consultation diffèrent.

ORACLE permet de consulter les données de deux façons :
- en cours d'exécution du prototype, on peut, en étant positionné sur un champ, exécuter une requête simple (touche QUERY) ou une requête plus complexe (touche QUERY WHERE avec spécification d'un critère par l'utilisateur)
- grâce à l'utilitaire UFI (User Friendly Interface) on peut, en dehors de l'exécution, consulter les données, de nouveau de façon simple ou complexe.

Les données consultées sont bien sûr les valeurs des champs (et avec UFI la structure des tables).

DSL-PROTO permet de consulter les données de plusieurs façons :
- en dehors de l'exécution, on peut consulter les occurrences des entités, relations, messages (et ainsi se rendre compte des modifications des données apportées par l'exécution du prototype)
- en cours d'exécution, il n'y a pas de consultation à proprement parler (sauf si l'affichage des données qu'on veut consulter est implémenté dans certaines fonctions).

Remarques :
- DSL-PROTO permet aussi de consulter les tables de la statique, les tables de la dynamique, qui sont extraites de la base de données des spécifications.
- De nombreux rapports documentaires permettent de présenter les résultats des consultations sous des formes variées (éventuellement graphiques).

B. MISE-A-JOUR DE LA BASE DE DONNEES

- ORACLE et DSL-PROTO permettent la mise-à-jour de la base de données en dehors de l'exécution de façon simple, ergonomique et rapide.
- En cours d'exécution, cette mise-à-jour est permise, mais de façon différente selon le logiciel :
* DSL-PROTO conduit l'utilisateur dans la mise-à-jour, et ce dernier ne peut effectuer que les mises-à-jour décidées et implémentées par le concepteur. Il y a donc un **contrôle rigoureux.**
* ORACLE permet à l'utilisateur de mettre-à-jour la base de données de façon subjective, volontaire, exceptés les contrôles effectués lors du COMMIT par les TRIGGERS (cfr chapitre 7).

Remarque : toutes les fonctions de mise-à-jour sont possibles : suppression, modification, ajout, moyennant respect des contraintes imposées quant à la structuration des données.

C. CREATION DE L'APPLICATION

Nous considérons que les deux logiciels, DSL-PROTO et ORACLE, permettent de créer l'application avec les mêmes résultats. Seules les formes de la création diffèrent (transformations nécessaires, architecture des traitements, ...).
Nous ne revenons pas sur ce point, qui a été largement développé lors de la présentation des deux logiciels (cfr chapitres 7 et 8).

Nous constatons que les deux logiciels ont permis de prototyper l'application "plans de pièces" dans son entièreté, en respectant les objectifs assignés aux différentes fonctions de l'analyse fonctionnelle, et en produisant les résultats attendus.

D. ECRITURE D'ALGORITHMES

Comme pour la création de l'application, nous pensons que les deux outils, DSL-PROTO et ORACLE, permettent de réaliser les objectifs assignés aux fonctions de l'application "plans de pièces".
Seule la manière diffère.

DSL-PROTO permet l'écriture d'algorithmes manipulant les données dans un langage (PROTO) très proche d'un langage de programmation traditionnel de haut niveau. Ces algorithmes obligent l'utilisateur à beaucoup de rigueur et de précision d'écriture, et ont l'avantage d'être très bien structurés et modulaires, permettant ainsi un développement très proche des techniques optimales de génie logiciel. On peut traduire un algorithme en "pseudo-langage" en algorithme PROTO sans difficulté.

ORACLE, outil disposant d'un langage SQL très performant, ne permet pas l'écriture d'algorithmes à proprement parler. L'utilisateur est obligé de disperser les différentes instructions de l'algorithme à réaliser en requêtes SQL de niveau champ ou de niveau fonction (cfr chapitre 7). Ces requêtes, dans leur ensemble, permettent de réaliser les mêmes primitives de manipulation de données que DSL-PROTO et de réaliser les mêmes traitements.
Nous n'avons éprouvé aucune difficulté à traduire les objectifs assignés aux fonctions prises dans leur ensemble, lors du prototypage avec ORACLE.

Les annexes 3 et 4 montrent que les différents types d'aide à l'élaboration des hypothèses que nous avions à implémenter (requêtes passives de niveau 1, 2, 3, calculs, contrôles, ...) ont pu être réalisées, que ce soit avec ORACLE ou DSL-PROTO.

E. EXECUTION

DSL-PROTO et ORACLE permettent d'exécuter plusieurs fois la même application sur les mêmes données.
En effet, l'utilisateur peut, avec DSL-PROTO, choisir s'il sauvegarde le résultat de l'exécution, et par conséquent s'il modifie ou non l'état de la base de données.
Quant à ORACLE, on peut faire toutes les modifications de données désirées, sans enregistrer celles-ci dans la base de données (pas de fonction COMMIT).
L'utilisateur peut donc apprendre à utiliser tout ou partie de l'application.

ORACLE et DSL-PROTO permettent aussi de commencer l'exécution à un certain endroit de la dynamique de l'application, soit par le système de menus ou de touches-fonctions, soit en choisissant le point d'entrée.

Nous détaillons au point 9.2.4. la comparaison des deux logiciels sur base des critères d'**exploitation** du prototypage.

CONCLUSION

Nous pouvons affirmer que globalement, DSL-PROTO et ORACLE ont les mêmes fonctionnalités dans le cadre du prototypage de l'application "plans de pièces".

9.2.2. LA FIABILITE DU PROTOTYPAGE

A partir des deux critères développés au point 9.1.2., évaluons et comparons DSL-PROTO et ORACLE.

A. LE RESPECT DES SPECIFICATIONS

Le respect des spécifications se situe à deux niveaux : les données et les traitements.

* au point de vue des données, et au point de vue des traitements, il est clair que DSL-PROTO est un logiciel **très fiable** quant au respect des spécifications, puisque le principe de DSL-PROTO est la **génération automatique** de la maquette à partir des spécifications. Celles-ci sont extraites de la base de données des spécifications par les différents modules de génération de la maquette.

* ORACLE est beaucoup moins fiable, dans la mesure où des transformations des spécifications doivent être effectuées par l'utilisateur afin de pouvoir générer la maquette. Il n'y a donc pas de génération automatique à partir des spécifications. **On ne peut donc pas prouver de manière automatisée que les spécifications sont respectées.**
Concernant les données, l'utilisateur doit faire la transformation du schéma entité-association en tables relationnelles.
Concernant les traitements, il y a quelques transformations à faire étant donné que les traitements en général, sont orientés écran.

B. LE CONTROLE DU RESPECT DES SPECIFICATIONS

En ce qui concerne le contrôle du respect des spécifications avec DSL-PROTO, il n'y a pas de problème, puisqu'un des résultats de l'exploitation du prototype est un fichier contenant la trace complète et précise de tous les "événements" : modification de la base de données (ajout d'entités, etc...), génération d'occurrences de messages, déclenchement de tel processus plutôt que tel autre en fonction de la valeur de telle condition, etc...
Ceci concerne donc DSL-PROTO proprement dit.
Un outil annexe, DSA (Dynamic Specification Analyser) permet de produire des rapports documentaires et d'interroger la base de données afin de permettre à l'utilisateur de vérifier si le prototypage a produit les résultats attendus sur les données.

Quant à ORACLE, il ne permet pas lui-même le contrôle du respect des spécifications.
L'utilisateur peut néanmoins vérifier, grâce à un outil annexe d'interrogation de la base de données, UFI (User Friendly Interface), si le prototypage a produit les résultats attendus sur les données.

9.2.3. LE DEVELOPPEMENT DU PROTOTYPE

Nous reprenons les trois critères développés au point 9.1.3., afin d'évaluer et comparer DSL-PROTO et ORACLE.

A. LA SIMPLICITE DE REALISATION

* Au niveau de la forme de la réalisation, on peut dire que :

1. Le logiciel ORACLE oblige l'utilisateur à répondre à beaucoup de questions, ce qui est un travail **long** et **fastidieux**, même si les réponses sont souvent courtes (Y/N).
De plus, un même champ de la base de données devra être décrit, dans le dialogue entre générateur d'application et concepteur, autant de fois qu'il est repris sur les différents écrans de l'application.
Le lecteur relira à cet effet les commentaires du point 7.5.

2. Avec DSL-PROTO, le concepteur génère son application grâce à un éditeur de texte classique.
Les seules difficultés proviennent , au point de vue de la forme, de la qualification non elliptique des éléments (cfr point 8.6.1.). Le résultat de la conception de l'application est contenu, avec DSL-PROTO, dans un fichier nettement plus petit et plus manipulable qu'avec ORACLE, ce qui est positif au point de vue de l'ergonomie du concepteur.

* Au niveau du contenu de la réalisation, nous pouvons :

- rappeler qu'en partant des spécifications, on génère plus facilement l'application avec DSL-PROTO qu'avec ORACLE, puisqu'il n'y a pas de transformations des données à faire.

- signaler que dans la mesure où le concepteur est familier, d'un côté aux notions de bases de données relationnelles (tables, clés, ...) et au langage SQL, d'un autre côté au langage DSL et aux primitives du langage PROTO (prototyping-procedure), il n'y a pas de réelle difficulté d'écriture.

B. EVOLUTIVITE

Nous envisagons l'évolutivité du prototype à deux niveaux : les modifications des structures des données et les modifications des traitements.

1. Les données

- Au niveau des modifications de la structure des données, ni ORACLE ni DSL-PROTO ne posent de problèmes importants.

- Au niveau des modifications des occurrences des données, ORACLE et DSL-PROTO possèdent un utilitaire de chargement permettant d'ajouter, modifier, supprimer des occurrences (champs de tables d'un côté, attributs d'entités et d'associations de l'autre), **et ce indépendamment de l'exécution de l'application.**

2. Les traitements

Il est clair que DSL-PROTO permet de modifier très facilement les traitements, puisque chaque fonction est bien définie, possède sa propre dynamique et sa propre statique.
Une modification de la dynamique des fonctions ou de la statique d'une fonction est facilement **localisable.**
De même, si le concepteur change l'objectif à assigner à une fonction, les modifications à apporter sont localisées à la procédure de la fonction.

ORACLE, de par son principe de construction par blocs et par écrans, peut poser certains problèmes au niveau des modifications des traitements.
Un exemple : chaque champ possède une position physique d'écran (page/ligne/colonne). Une modification de la dynamique peut amener à devoir changer manuellement les spécifications des positions physiques de tous les champs affichés.
Cette évolutivité difficile d'ORACLE quant aux traitements découle bien sûr des transformations préalables qu'il exige par rapport aux résultats de l'analyse fonctionnelle.

C. MODULARITE

Signalons d'abord qu'ainsi que nous avons pu le remarquer au point B. ci-dessus, la modularité offerte par DSL-PROTO lui permet d'être plus évolutif.
Il est clair que pour le concepteur, DSL-PROTO offre une grande modularité. En effet,
1. un processus peut être extrait de la dynamique d'une phase et participer à la dynamique d'une autre phase sans modification (sauf éventuellement les noms. A ce sujet, les procédures avec paramètres qui seront ultérieurement implémentées présenteront beaucoup d'avantages).
2. une phase par exemple peut être insérée entre la phase qui la précède et celle qui la suit conceptuellement. Ceci ne pose aucun problème.
On peut imaginer que la dernière fonction de la phase (i-1) déclenche, en terminaison, la première fonction de la phase (i). La dernière fonction de celle-ci déclenche, en terminaison, la première fonction de la phase (i+1).
3. Au point de vue des données, le modèle de la structuration des données permet toutes les modularisations possibles, grâce à la notion de sous-schéma conceptuel des données.

Si la modularité de ORACLE est évidente au point de vue des données, elle l'est moins pour les traitements, de par le principe de conception de l'application : une application porte un nom, les noms des champs sont préfixés par le nom d'un bloc, les pages d'écran sont numérotées de manière absolue, etc... Nous ne développons pas plus précisemment ce point.

9.2.4. L'EXPLOITATION DU PROTOTYPE

Nous évaluons et comparons les quatre critères développés au point 9.1.4.

A. FACILITE D'EXPLOITATION

Au niveau de l'exploitation, ORACLE offre un très bon niveau d'accessibilité aux non-informaticiens. L'utilisateur, lorsqu'il connait le principe des menus et des touches-fonctions, quelques notions sur les tables et les bases de données (création d'un record, COMMIT, etc...) peut exécuter le prototype.
Celui-ci est d'autant plus facile à exploiter que les écrans sont clairs, nets, bien définis, que l'utilisateur est aidé dans ses choix par les menus, que des messages soit d'erreur, soit de signalisation d'actions effectivement réalisées apparaissent de manière lisible. Il est clair que l'utilisateur peut voir par quoi seront remplacées certaines tâches ardues, longues et répétitives.
Le lecteur peut constater la facilité d'exploitation d'ORACLE en examinant les écrans présentés à l'annexe 2.

Nous ne pouvons juger que difficilement de la facilité d'exploitation de DSL-PROTO, dans la mesure où le gestionnaire d'écran, qui est actuellement implémenté, n'était pas encore intégré au moment où nous avons effectué les tests.

B. ERGONOMIE

Nous pensons qu'ORACLE offre toutes les garanties d'ergonomie à l'utilisateur final, qui est surtout intéressé par la cinématique des écrans. On peut reprendre à ce sujet les mêmes considérations que celles développées ci-dessus au point A.

L'ergonomie de DSL-PROTO est moindre pour l'utilisateur final, même avec intégration du gestionnaire d'écran : il n'y a pas de touches-fonctions, il n'y a pas d'aide à l'exécution sous forme de messages d'erreur ou de messages explicitant le travail à effectuer.

C. TEMPS DE REPONSE

ORACLE prend en considération le problème du temps de réponse, étant donné sa finalité de Système de Gestion de Bases de Données.
Nous avons eu l'occasion de tester le prototypage de l'application "plans de pièces" sur la base de données réelle, de taille élevée.
Les possibilités de gestion d'index offertes par ORACLE permettent d'obtenir des temps de réponse que les utilisateurs et les concepteurs ont jugés bons.

Nous ne pouvons pas préjuger des temps de réponse avec DSL-PROTO, étant donné que ce logiciel est appelé à travailler sur une base de données de taille limitée et ne prend donc pas les temps de réponse en considération.
Néanmoins, les problèmes de prototypage avec DSL-PROTO sous une base de données relationnelles sont étudiés dans [PEN-86].

D. CONFIDENTIALITE

ORACLE, par la pose de permissions sur les données d'une part, par la limitation de l'autorisation d'exécuter l'application d'autre part (l'utilisateur doit donner nom et mot de passe), permet de gérer la confidentialité des données et des traitements.

Cet aspect n'est pas implémenté dans le logiciel DSL-PROTO.

CONCLUSION GENERALE

Ce mémoire concerne le prototypage du processus de conception de la filière mécanique d'un véhicule automobile.

Le titre du mémoire est significatif quant à la présence de deux axes d'études.

Le premier axe concerne l'étude du processus de conception de la filière mécanique d'un véhicule automobile.
C'est ce que nous avons réalisé dans les deux premières parties.

Le second axe repose sur le premier et présente l'étude du prototypage du processus de conception de la filière mécanique d'un véhicule automobile, du moins une partie de ce processus.
C'est l'objet de la troisième partie.

Dans la première partie, nous avons décrit de façon générale le processus de conception de la filière mécanique d'un véhicule automobile, ainsi que le système d'information qui en est le support.
Nous avons aussi présenté le problème à résoudre, en nous attachant notamment à un point particulier et important : l'aide à l'élaboration des hypothèses, qui est omniprésente, sous diverses formes.

La deuxième partie s'applique à un sous-ensemble de la totalité du problème à résoudre : l'application "plans de pièces" du projet "Filière Méthodes Mécaniques" développé dans la première partie.
Dans ce but, nous avons présenté le schéma conceptuel de l'application "plans de pièces". Nous avons d'abord développé le schéma conceptuel des données, dont la réalisation, lors du stage à la Régie Nationale des Usines Renault (Paris) a été un long travail, tant d'une part, l'acquisition des connaissances et la complexité des données à modéliser étaient importantes, tant d'autre part les problèmes méthodologiques soulevés et découverts lors de l'élaboration de ce schéma conceptuel n'étaient pas faciles à résoudre et ont retenu toute notre attention. Il est important de le souligner.
Aussi avons-nous consacré un chapitre à la justification méthodologique de ce schéma conceptuel des données.
Pour terminer la deuxième partie, nous avons présenté le schéma conceptuel des traitements. La dynamique des phases et la dynamique des fonctions sont justifiées et commentées au point de vue méthodologique.

Nous avons aussi voulu montrer, dans cette deuxième partie, en quelle manière les fonctions résultant de l'analyse fonctionnelle permettent d'implémenter l'aide à l'élaboration des hypothèses.

Dans la troisième partie, nous avons développé l'étude du prototypage de l'application "plans de pièces".

Nous avons d'abord présenté le prototypage : définitions, motivations, ainsi que les caractéristiques du prototypage de l'application "plans de pièces" : caractéristiques générales, de conception et d'utilisation. Nous avons spécifié les contraintes concernant l'outil informatique dans le cadre de ce prototypage.

Afin de mettre en oeuvre le prototypage de l'application "plans de pièces", nous avons présenté brièvement deux outils de prototypage rapide : ORACLE, Système de Gestion de Bases de Données, et DSL-PROTO, logiciel de prototypage de l'atelier IDA (Interactive Design Approach).
Nous avons montré comment l'application pouvait être prototypée à l'aide de ces deux outils. Nous avons montré les fonctions à réaliser, les transformations éventuelles des spécifications à effectuer en fonction de l'outil utilisé, que ces transformations portent sur les données ou sur les traitements.

Nous avons terminé en évaluant et comparant les deux outils, ORACLE et DSL-PROTO, suivant quatre grands types de critères : les fonctionnalités, la fiabilité, le développement et l'exploitation du prototypage.
Ainsi que nous l'avions dit en introduction du dernier chapitre, nous n'avons pas voulu comparer systématiquement les deux logiciels et préférer l'un à l'autre, mais plutôt les évaluer et montrer leurs avantages et inconvénients suivant différents axes.
Nous avons vu que l'objectif assigné au prototypage avait une grande importance lors de l'évaluation : prototypage à destination de l'utilisateur final ou prototypage à destination du concepteur.

Nous voulons terminer ce mémoire en posant la réflexion suivante :
DSL-PROTO est un outil de prototypage par génération automatique du programme à partir des spécifications fonctionnelles décrites suivant la méthodologie d'analyse fonctionnelle développée principalement dans [BOD-83].
ORACLE est avant tout un Système de Gestion de Bases de Données, avec lequel la génération d'une application ne découle pas de manière automatique des spécifications.
Nous pouvons nous demander dans quelle mesure nous ne favorisons pas, en partant des résultats de l'analyse fonctionnelle pour évaluer et comparer les deux logiciels, le logiciel dont le prototype est généré à partir des spécifications, dans quelle mesure nous ne favorisons pas DSL-PROTO.

En d'autres termes, on peut se demander quel est le rôle de la deuxième partie (schéma conceptuel) dans le cadre du prototypage de l'application "plans de pièces" avec le logiciel ORACLE, on

peut se demander si les spécifications fonctionnelles présentées dans la deuxième partie sont la meilleure base pour la mise en oeuvre d'un prototypage rapide avec un S.G.B.D. relationnel.

Poser la question, c'est en quelque sorte avoir envie d'y répondre. Mais ne jugeons pas trop vite. Avant de pouvoir affirmer une position, positivement ou négativement, il serait nécessaire et intéressant de développer le prototypage avec ORACLE sur base de spécifications exprimées selon un autre formalisme que celui de la méthodologie IDA.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[BLI-86] : Michel BLIN, "IDA", dans la revue "Génie logiciel", numéro 3, 1986.

[BOD-83] : François BODART et Yves PIGNEUR, "Conception assistée des applications informatiques. 1. Etude d'opportunité et analyse conceptuelle", P.U.N., 1983.

[BRO-82] : M.L. BRODIE et E. SILVA, "Active and passive component modelling : ACM/PCM", dans "Information Systems Design Methodologies : a comparative review", T.W. OLLE, H.G. FOL, A.A. VERRIJN-STUART, 1982.

[CHO-86] : Christine CHOPPY, "Techniques et aspects du prototypage", dans la revue "Génie logiciel", numéro 3, 1986.

[DSA-84] : Manuel de référence de DSA, atelier logiciel IDA, 1984.

[DUC-86] : Jacques DUCLOY, "Définitions du prototypage", dans la revue "Génie logiciel", numéro 3, 1986.

[FLO-85] : FLOYD, "Maquettes et prototypes informatiques : état de l'art et perspectives", dans "le Monde Informatique", avril 1985.

[FUR-86] : A.L. FURTADO, E.J. NEUHOLD, "Formal Techniques for Database Design", 1986.

[HAI-86] : J.L. HAINAUT, "Conception Assistée des Applications Informatiques. 2. Conception de la base de données.", P.U.N., 1986.

[LEG-86] : Bruno LEGEARD, "Définitions du prototypage", dans la revue "Génie logiciel", numéro 3, 1986.

[LEG-2-86] : Armelle LE GUEVEL, "Prototypage, complément de la modélisation", dans la revue "Génie logiciel", numéro 3, 1986.

[ORA-1-84] : ORACLE : Interactive Application Facility : Application Designer's Guide, 1984.

[ORA-2-84] : ORACLE : Interactive Application Facility: Terminal Operator's Guide, 1984.

[ORA-3-84] : ORACLE : Interactive Application Facility : Application Designer's Reference Manual, 1984.

[PEN-86] : Dominique PENNINCKX, Mémoire : "Gestion du système de prototypage DSL-PROTO sous une base de données relationnelle", 1986.

[PRO-86] : Manuel de référence de DSL-PROTO, 1986. (version non définitive).

[RAU-86] : Jean-Claude RAULT, éditorial de la revue "Génie logiciel", numéro 3, 1986.

[REN-1-85] : Document Renault : "Présentation du produit ORACLE", Service D.S.C.I.T., par N. LECLAIR, E. GHESQUIERE, C. SENE, G. DESCLAVELIERES, H. PHAM, juin 1985.

[REN-2-85] : Document Renault : "Projet Filière Méthodes Mécaniques : Dossier Pour Investissements", par N. LECLAIR et G. BARAT, 1985.

LISTE DES FIGURES

LISTE DES FIGURES

ANNEXES

ANNEXE 1 : DESCRIPTION DU SCHEMA CONCEPTUEL DES DONNEES

Le schéma conceptuel des données est décrit dans le langage DSL.

1. DEFINITION DES ENTITES

```
DEFINE ENTITY boite-v-dmc ;
DESCRIPTION ;  boîte de vitesses dont les caractéristiques sont
               déterminées par la DMC ou ont une utilité pour la
               DMC dans l'élaboration des plans de pièces.
               Contrainte d'existence : si niveau-org-dmc =
               'sous-famille-organe' ou 'organe', une occurrence
               de boite-v-dmc n'existe que si elle participe à une
               seule occurrence de la relation regroupement-
               boite-v ;
RELATED BY commerce-bv-dmc, correspondance-boite-v, decom-bv-pie-
           dmc, decom-veh-bv-dmc, montage-boite-v, regroupement-
           boite-v ;
CONSISTS OF id-bv, description, niveau-org-dmc, coef-loupe,
            coef-mpr ;
IDENTIFIED BY id-bv ;

DEFINE ENTITY boite-v-dpi ;
DESCRIPTION ;  boîte de vitesses dont les caractéristiques sont
               déterminées par la DPI.
               Contrainte d'existence : une occurrence ne peut
               exister que si elle participe à au moins une
               occurrence de la relation decom-veh-bv-dpi ;
RELATED BY commerce-bv-dpi, correspondance-boite-v, decom-veh-
           bv-dpi ;
CONSISTS OF description, id-bv ;
IDENTIFIED BY id-bv ;

DEFINE ENTITY chassis-dmc ;
DESCRIPTION ;  châssis dont les caractéristiques sont déterminées
               par la DMC ou ont une utilité pour la DMC dans
               l'élaboration des plans de pièces.
               Contrainte d'existence : cfr boite-v-dmc ;
RELATED BY commerce-cha-dmc, correspondance-chassis, decom-cha-
           pie-dmc, decom-veh-cha-dmc, montage-chassis, regrou-
           pement-chassis ;
CONSISTS OF id-cha, description, niveau-org-dmc, coef-loupe,
            coef-mpr ;
IDENTIFIED BY id-cha ;

DEFINE ENTITY chassis-dpi ;
```

```
DESCRIPTION ;  châssis dont les caractéristiques sont déterminées
               par la DPI.
               Contrainte d'existence : cfr boite-v-dpi ;
RELATED BY  commerce-cha-dpi,  correspondance-chassis,  decom-veh-
            cha-dpi ;
CONSISTS OF id-cha, description ;
IDENTIFIED BY id-cha ;

DEFINE ENTITY hypothese ;
DESCRIPTION ;  essai de correspondance DPI-DMC,  de nomenclature,
               de  traduction  des besoins,  de  plans de pièces,
               fait par la DMC ;
CONSISTS OF num-hypo ;
IDENTIFIED BY num-hypo ;

DEFINE ENTITY ligne-product ;
DESCRIPTION ; ligne de montage d'organes ou d'usinage de pièces ;
RELATED  BY   montage-boite-v,  montage-chassis,  montage-moteur,
              usinage ;
CONSISTS OF nom-ligne, capacite-a-terme, 14 echeancier-a-terme ;
IDENTIFIED BY nom-ligne ;


DEFINE ENTITY moteur-dmc ;
DESCRIPTION ;  moteur dont les  caractéristiques sont déterminées
               par  la  DMC ou ont  une utilité pour  la DMC dans
               l'élaboration des plans de pièces.
               Contrainte d'existence : cfr boite-v-dmc ;
RELATED  BY  commerce-mot-dmc,  correspondance-moteur,  decom-mot-
             pie-dmc,  decom-veh-mot-dmc,  montage-moteur,  regrou-
             pement-moteur ;
CONSISTS  OF  id-mot,  description,  niveau-org-dmc,  coef-loupe,
              coef-mpr ;
IDENTIFIED BY id-mot ;

DEFINE ENTITY moteur-dpi ;
DESCRIPTION ;  moteur dont les  caractéristiques sont déterminées
               par la DPI.
               Contrainte d'existence : cfr boite-v-dpi ;
RELATED  BY  commerce-mot-dpi,  correspondance-moteur,  decom-veh-
             mot-dpi ;
CONSISTS OF id-mot, description ;
IDENTIFIED BY id-mot ;

DEFINE ENTITY pays ;
DESCRIPTION ;  lieu de  commercialisation de  véhicules  ou d'or-
               ganes ;
RELATED  BY  commerce-bv-dmc,  commerce-bv-dpi,  commerce-cha-dmc,
             commerce-cha-dpi,   commerce-mot-dmc,   commerce-mot-
             dpi,  commerce- pie-dmc,  commerce-mot-dpi ;
```

CONSISTS OF nom-pays, zone-pays ;
IDENTIFIED BY nom-pays ;

DEFINE ENTITY piece-dmc ;
DESCRIPTION ; pièce dont les caractéristiques sont déterminées par la DMC ou ont une utilité pour la DMC dans l'élaboration des plans de pièces.
Contrainte d'existence : si niveau-pie-dmc = 'type-piece' ou 'piece', une occurrence de piece-dmc n'existe que si elle participe à une seule occurrence de la relation regroupement-piece ;
RELATED BY commerce-pie-dmc, composition-pieces, decom-bv-pie-dmc, decom-cha-pie-dmc, decom-mot-pie-dmc, regroupement-piece, usinage ;
CONSISTS OF id-pie-dmc, description, niveau-pie-dmc, coef-mpr, coef-loupe ;
IDENTIFIED BY id-pie-dmc ;

DEFINE ENTITY vehicule-dmc ;
DESCRIPTION ; véhicule dont les caractéristiques sont déterminées par la DMC ou ont une utilité pour la DMC dans l'élaboration des plans de pièces.
Contrainte d'existence : une occurrence de vehicule-dmc ne peut exister que si elle participe à au moins une occurrence des relations decom-veh-bv-dmc, decom-veh-cha-dmc, decom-veh-mot-dmc ;
RELATED BY affectation-version, decom-veh-bv-dmc, decom-veh-cha-dmc, decom-veh-mot-dmc, regroupement-vehicule ;
CONSISTS OF id-veh, description, niveau-veh-dmc ;
IDENTIFIED BY id-veh ;

DEFINE ENTITY vehicule-dpi ;
DESCRIPTION ; véhicule dont les caractéristiques sont déterminées par la DPI.
Contrainte d'existence : une occurrence de vehicule-dpi ne peut exister que si elle participe à une seule occurrence des relations decom-veh-mot-dpi, decom-veh-cha-dpi, decom-veh-bv-dpi ;
RELATED BY commerce-veh-dpi, decom-veh-bv-dpi, decom-veh-cha-dpi, decom-veh-mot-dpi ;
CONSISTS OF mod-veh-dpi, denom-veh-dpi, date-d-f, date-f-f, dir-assistee, observations, version-veh-dpi ;
IDENTIFIED BY (mod-veh-dpi, version-veh-dpi) ;

2. DEFINITION DES RELATIONS

```
DEFINE RELATION commerce-bv-dmc ;
DESCRIPTION   ;   commercialisation   d'une   boîte   de   vitesses
                  caractérisée par la DMC dans un pays ;
                  Connectivités :
                  .   un pays peut exister sans commercialiser de
                  boîte de vitesses,  et il peut en commercialiser
                  plusieurs ;
                  .  une boîte de vitesses peut exister sans avoir
                  de pays où elle est commercialisée et  elle peut
                  être commercialisée dans plusieurs pays ;
RELATES pays WITH CONNECTIVITY 0-N ;
RELATES boite-v-dmc WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;
CONSISTS OF 14 besoins-dyn, int-locale ;

DEFINE RELATION commerce-bv-dpi ;
DESCRIPTION ;  commercialisation d'une boîte de vitesses caracté-
               risée par la DPI dans un pays.
               Connectivités :
               .   une  boîte de  vitesses-dpi  peut exister sans
               qu'on  sache dans  quel pays la  commercialiser et
               elle peut être commercialisée dans plusieurs pays
               .   un pays  peut  exister sans  commercialiser de
               boîtes, et il peut en commercialiser plusieurs ;
RELATES boite-v-dpi WITH CONNECTIVITY 0-N ;
RELATES pays WITH CONNECTIVITY 0-N ;
CONSISTS OF 14 besoins-dpi ;

DEFINE RELATION commerce-cha-dmc ;
cfr commerce-bv-dmc ;

DEFINE RELATION commerce-cha-dpi ;
cfr commerce-bv-dpi ;

DEFINE RELATION commerce-mot-dmc ;
cfr commerce-bv-dmc ;

DEFINE RELATION commerce-mot-dmc ;
cfr commerce-bv-dpi ;

DEFINE RELATION commerce-pie-dmc ;
cfr commerce-bv-dmc ;

DEFINE RELATION commerce-veh-dpi ;
cfr commerce-bv-dpi ;
```

```
DEFINE RELATION composition-pieces ;
DESCRIPTION ;  composition d'une pièce caractérisée par la DMC en
               d'autres pièces caractérisées par la DMC.
               Connectivités :
               .  une pièce peut n'être composée d'aucune autre
               pièce et elle peut être composée de plusieurs
               autres pièces
               .  une pièce peut exister sans être composante
               d'une autre pièce et elle peut être composante de
               plusieurs autres pièces
               Contrainte d'existence : une occurrence de la
               relation composition-pieces ne peut exister que
               si niveau-pie-dmc = 'type-piece' pour la piece
                  composée et 'piece' pour la pièce composante
               ou si niveau-pie-dmc = 'famille-piece' pour la
                     pièce composée et 'type-piece' ou 'piece'
                     pour la pièce composante
               ou si niveau-pie-dmc = 'piece' pour la pièce
                     composée et la pièce composante
RELATES piece-dmc AS "composé-de" WITH CONNECTIVITY 0-N ;
RELATES piece-dmc AS "composant-de"WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-N ;
CONSISTS OF quantité-composition ;

DEFINE RELATION correspondance-boite-v ;
DESCRIPTION ; correspondance entre une boîte de vitesses caracté-
              risée par la DPI et une boîte de vitesses
              caractérisée par la DMC.
              Connectivités :
              .  une boîte de vitesses du descriptif peut n'avoir
              aucun correspondant dans la nomenclature et elle
              peut avoir plusieurs boîtes de vitesses lui
              correspondant dans la nomenclature
              .  une boîte de vitesses-DMC peut ne correspondre à
              aucune boîte du descriptif et elle peut
              correspondre à plusieurs boîtes du descriptif
RELATES boite-v-dpi WITH CONNECTIVITY 0-N ;
RELATES boite-v-dmc WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;

DEFINE RELATION correspondance-chassis ;
cfr correspondance-boite-v ;

DEFINE RELATION correspondance-moteur ;
cfr correspondance-boite-v ;

DEFINE RELATION decom-bv-pie-dmc ;
```

DESCRIPTION ; décomposition d'une boîte de vitesses caractérisée par la DMC en pièces caractérisée par la DMC .
Connectivités :
. dans la nomenclature, une boîte de vitesses peut exister sans qu'on lui ait attribué des pièces, et elle peut être composée de plusieurs pièces
. une pièce peut exister sans faire partie d'une boîte de vitesses et elle peut faire partie de plusieurs boîtes de vitesses
Contrainte d'existence : dans le cas où niveau-pie -dmc = 'type-piece' ou 'famille-piece', une occurrence de decom-bv-pie-dmc n'existe que si niveau-org-dmc de boite-v-dmc = 'sous-famille-organe' ou 'organe'
RELATES boite-v-dmc WITH CONNECTIVITY 0-N ;
RELATES piece-dmc WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;
CONSISTS OF coef-pie-dmc ;

DEFINE RELATION decom-cha-pie-dmc ;
cfr decom-bv-pie-dmc ;

DEFINE RELATION decom-mot-pie-dmc ;
cfr decom-bv-pie-dmc ;

DEFINE RELATION decom-veh-bv-dmc ;
DESCRIPTION ; décomposition d'un véhicule caractérisé par la DMC en une boîte de vitesses caractérisée par la DMC.
Connectivités :
. un véhicule-DMC est composé d'au moins une boîte de vitesses et il peut avoir plusieurs décompositions possibles
. une boîte de vitesses-DMC peut exister sans être affectée à un véhicule-DMC ; on peut retrouver la même boîte de vitesses DMC sur plusieurs véhicules-DMC.
Contrainte d'existence : une occurrence de la relation decom-veh-bv-dmc n'existe que si niveau-veh-dmc = 'version'.
RELATES vehicule-dmc WITH CONNECTIVITY 1-N ;
RELATES boite-v-dmc WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;

DEFINE RELATION decom-veh-bv-dpi ;
DESCRIPTION ; décomposition d'un véhicule caractérisé par la DPI en une boîte de vitesses caractérisée par la DPI.
Connectivités :
. un véhicule-DPI est composé d'une et une seule boîte de vitesses-DPI

. une boîte de vitesses-DPI n'est définie que lorsqu'elle fait partie d'un véhicule-DPI, et elle peut faire partie de plusieurs véhicules-DPI.

RELATES vehicule-dpi WITH CONNECTIVITY 1-1 ;
RELATES boite-v-dpi WITH CONNECTIVITY 1-N ;

DEFINE RELATION decom-veh-cha-dmc ;
cfr decom-veh-bv-dmc ;

DEFINE RELATION decom-veh-cha-dpi ;
cfr decom-veh-bv-dpi ;

DEFINE RELATION decom-veh-mot-dmc ;
cfr decom-veh-bv-dmc ;

DEFINE RELATION decom-veh-mot-dpi ;
cfr decom-veh-bv-dpi ;

DEFINE RELATION montage-boite-v ;
DESCRIPTION ; montage d'une boîte de vitesses caractérisée par la DMC sur une ligne de production.
Connectivités :
. une boîte de vitesses-DMC peut exister sans qu'on sache sur quelle ligne de production la monter ; elle peut être montée sur plusieurs lignes de production.
. une ligne de production peut exister sans monter de boîte de vitesses et elle peut en monter plusieurs.
Contrainte d'exclusion : la participation d'une occurrence de ligne-product dans une occurrence de montage-boite-v exclut sa participation dans une occurrence des relations montage-chassis, montage-moteur, usinage .

RELATES boite-v-dmc WITH CONNECTIVITY 0-N ;
RELATES ligne-product WITH CONNECTIVITY 0-N ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;
CONSISTS OF 14 engagements ;

DEFINE RELATION montage-chassis ;
cfr montage-boite-v ;

DEFINE RELATION montage-moteur ;
cfr montage-boite-v ;

DEFINE RELATION regroupement-boite-v ;

```
DESCRIPTION ;  regroupement pour une boîte de vitesses DMC, entre
               ses différents niveaux de raffinement : organe,
               sous-famille- organe, famille-organe.
               Connectivités :
               .  une famille (une sous-famille) de boîtes de
               vitesses peut exister sans qu'on sache les
               sous-familles (les organes) qui la raffinent et
               elle peut être raffinée en plusieurs sous-familles
               (organes).
               .  une famille est au niveau supérieur de raffine-
               ment.  Elle ne peut pas être regroupée. Une
               sous-famille (un organe) est regroupée dans une
               seule famille (sous-famille).
RELATES boite-v-dmc AS "regroupe" WITH CONNECTIVITY 0-N ;
RELATES boite-v-dmc AS "est regroupé" WITH CONNECTIVITY 0-1 ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;

DEFINE RELATION regroupement-chassis ;
cfr regroupement-boite-v ;

DEFINE RELATION regroupement-moteur ;
cfr regroupement-boite-v ;

DEFINE RELATION regroupement-piece ;
DESCRIPTION ;  regroupement pour une pièce DMC, entre ses
               différents niveaux de raffinement : piece,
               type-piece, famille-piece.
               Connectivités :
               .  une famille (un type) de pièces peut exister
               sans qu'on sache les types (les pièces) qui la
               raffinent et elle peut être raffinée en plusieurs
               types (pièces).
               .  une famille est au niveau supérieur de raffine-
               ment.  Elle ne peut pas être regroupée. Un type
               (une pièce) est regroupé dans une seule famille
               (type).
RELATES piece-dmc AS "regroupe" WITH CONNECTIVITY 0-N ;
RELATES piece-dmc AS "est regroupé" WITH CONNECTIVITY 0-1 ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;

DEFINE RELATION regroupement-vehicule ;
DESCRIPTION ;  regroupement pour un véhicule DMC, entre ses
               différents niveaux de raffinement : gamme,
               famille, version.
               Connectivités :
               .  une gamme (une famille) de véhicules peut
               exister sans qu'on sache les familles (les
               versions) qui la raffinent et elle peut être
               raffinée en plusieurs familles (versions).
```

. une gamme est au niveau supérieur de raffinement. Elle ne peut pas être regroupée. Une famille (une version) est regroupée dans une seule gamme (famille).

```
RELATES vehicule-dmc AS "regroupe" WITH CONNECTIVITY 0-N ;
RELATES vehicule-dmc AS "est regroupé" WITH CONNECTIVITY 0-1 ;
RELATES hypothese WITH CONNECTIVITY 0-1 ;

DEFINE RELATION usinage ;
cfr montage-boite-v ;
```

3. DEFINITION DES GROUPES

```
DEFINE GROUP code-bv ;
DESCRIPTION ;  code d'une boîte de  vitesses DPI  ou DMC, faisant
               partie de l'identifiant de celle-ci ;
CONSISTS OF famille-bv, sous-famille-bv ;

DEFINE GROUP code-mot ;
DESCRIPTION ;  code d'un  moteur  DPI ou  DMC,  faisant partie de
               l'identifiant de celui-ci ;
CONSISTS OF famille-mot, sous-famille-mot, option-mot ;

DEFINE GROUP id-bv ;
DESCRIPTION ;  codification identifiante d'une boîte  de vitesses
               DPI ou DMC ;
CONSISTS OF code-bv, ind-bv ;

DEFINE GROUP id-mot ;
DESCRIPTION ; codification identifiante d'un moteur DPI ou DMC ;
CONSISTS OF code-mot, ind-mot ;

DEFINE GROUP id-pie-dmc ;
DESCRIPTION ; codification identifiante d'une pièce DMC ;
CONSISTS OF code-pie-dmc, code-type-pie-dmc, code-fam-pie-dmc ;

DEFINE GROUP id-veh ;
DESCRIPTION ; identifiant d'un véhicule DMC ;
CONSISTS  OF  code-gamme-dmc,  modele-famille-dmc, version-codee-
              veh-dmc, zone-geogr-veh-dmc ;

DEFINE GROUP ind-bv ;
DESCRIPTION ;  indice d'une boîte de vitesses,  faisant partie de
               l'identifiant de celle-ci ;
CONSISTS OF car-pour-bv, mot-pour-bv, gamme-veh-pour-bv ;

DEFINE GROUP ind-mot ;
DESCRIPTION ; indice d'un moteur, faisant partie de l'identifiant
              de celui-ci ;
CONSISTS OF car-pour-mot, bv-pour-mot ;

DEFINE GROUP version-veh-dpi ;
DESCRIPTION  ;  code de  la  version  du  véhicule-DPI  tel qu'il
                apparait dans le descriptif ;
CONSISTS OF car-v-v-dpi,  en-v-v-dpi, mt-v-v-dpi, bv-v-v-dpi, eq-
            v-v-dpi, dis-v-v-dpi ;
```

4. DESCRIPTION DES ELEMENTS

```
DEFINE ELEMENT bv-pour-mot ;
DESCRIPTION ;  codification de la boîte de vitesses à utiliser en
               relation avec un moteur défini ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT bv-v-v-dpi ;
DESCRIPTION ;  codification de la boîte de vitesses de la version
du véhicule DPI telle qu'elle apparait dans le descriptif ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT capacite-a-terme ;
DESCRIPTION ; ce que peut fabriquer au maximum une ligne ;
FORMAT IS "int(4)" ;

DEFINE ELEMENT car-pour-bv ;
DESCRIPTION  ;  codification  de  la  carrosserie  à  utiliser en
               relation avec la boîte de vitesses ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT car-pour-mot ;
DESCRIPTION  ;  codification  de  la  carrosserie  à  utiliser en
               relation avec le moteur ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT car-v-v-dpi ;
DESCRIPTION ;  codification de  la  carrosserie de  la version du
               véhicule  DPI  telle  qu'elle  apparait  dans  le
               descriptif ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT code-fam-pie-dmc ;
DESCRIPTION ;  codification identifiante d'une famille  de pièces
               DMC ;
FORMAT IS "alpha(15)" ;

DEFINE ELEMENT code-gamme-dmc ;
DESCRIPTION ;  codification d'une gamme de véhicule  DMC, faisant
               partie de l'identifiant du véhicule DMC ;
FORMAT IS "alpha(5)" ;

DEFINE ELEMENT code-pie-dmc ;
DESCRIPTION ; codification identifiante d'une pièce DMC ;
FORMAT IS "alpha(10)" ;

DEFINE ELEMENT code-type-pie-dmc ;
DESCRIPTION ; codification identifiante d'un type de pièces DMC ;
```

```
FORMAT IS "alpha(20)" ;

DEFINE ELEMENT coef-loupe ;
DESCRIPTION ;  pourcentage de la somme [pour la commercialisation
               +  pour le MPR] d'organes DMC ou  de  pièces DMC à
               monter (ou à usiner)  en plus pour  faire face aux
               erreurs de fabrication .
               Contrainte :  différent  de  "valeur  inconnue" si
                             niveau-org-dmc  <> "famille-organe"
                             pour un organe ou si niveau-pie-dmc
                             <> "famille-piece" pour une pièce.
FORMAT IS "int(3).dec(2)" ;

DEFINE ELEMENT coef-mpr ;
DESCRIPTION ;  pourcentage  du  nombre  d'organes ou  de pièces à
                produire en plus pour le MPR ;
                Contrainte  :  différent de "valeur  inconnue" si
                               niveau-org-dmc  <> "famille-organe"
                               pour un organe ou si niveau-pie-dmc
                               <> "famille-piece" pour une pièce.
FORMAT IS "int(3).dec(2)" ;

DEFINE ELEMENT coef-pie-dmc ;
DESCRIPTION ; quantité d'une pièce DMC dans un organe DMC ;
FORMAT IS "int(3)" ;

DEFINE ELEMENT date-d-f ;
DESCRIPTION ;  date de  début de  fabrication  de  la  version du
                véhicule  DPI,  telle  qu'elle  apparait  dans le
                descriptif ;
FORMAT IS "int(4)" ;

DEFINE ELEMENT date-f-f ;
DESCRIPTION  ;  date de  fin  de  fabrication  de  la  version du
                véhicule  DPI,  telle  qu'elle  apparait  dans le
                descriptif ;
FORMAT IS "int(4)" ;

DEFINE ELEMENT denom-veh-dpi ;
DESCRIPTION ;  appellation du véhicule-DPI telle qu'elle apparait
               dans le descriptif ;
FORMAT IS "alpha(12)" ;

DEFINE ELEMENT description ;
DESCRIPTION  ;  description  d'un  organe,  d'un  véhicule, d'une
                pièce, ... ;
FORMAT IS "alpha(30)" ;

DEFINE ELEMENT dir-assistee ;
```

```
DESCRIPTION  ;   codification permettant de déterminer si la
                 direction assistée d'une version de véhicule DPI
                 est optionnelle ou montée sur toute la série ;
DOMAIN OF VALUE ARE "ser","opt" ;
FORMAT IS "alpha(3)" ;

DEFINE ELEMENT dis-v-v-dpi ;
DESCRIPTION ;  codification du niveau de pollution de la version
               de véhicule DPI telle qu'elle apparait dans le
               descriptif ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT en-v-v-dpi ;
DESCRIPTION ;  codification de l'énergie caractériant la version
               de véhicule DPI telle qu'elle apparait dans le
               descriptif ;
DOMAIN OF VALUE ARE "e","d" ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT eq-v-v-dpi ;
DESCRIPTION  ;  codification de l'équipement caractérisant la
                version de véhicule DPI telle qu'elle apparait
                dans le descriptif ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT famille-bv ;
DESCRIPTION ;  codification de la famille de boîte de vitesses
                telle qu'elle apparait dans le code d'une boîte
                de vitesses ;
FORMAT IS "alpha(2)" ;

DEFINE ELEMENT famille-mot ;
DESCRIPTION ;  codification de la famille de moteur telle qu'elle
               apparait dans le code d'un moteur ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT gamme-veh-pour-bv ;
DESCRIPTION ;  codification de la gamme de véhicule à utiliser en
               relation avec la boîte de vitesses ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT id-cha ;
DESCRIPTION ; codification identifiante d'un châssis ;
FORMAT IS "alpha(6)" ;

DEFINE ELEMENT int-locale ;
DESCRIPTION ;  quantité, partie des besoins d'un organe ou d'une
               pièce pour un pays fabriquée, usinée dans ce pays;
               on considère cette quantité constante sur le plan
               septennal ;
```

```
FORMAT IS "int(8)" ;

DEFINE ELEMENT mod-veh-dpi ;
DESCRIPTION ;  code du modèle de véhicule-DPI tel  qu'il apparait
               dans le descriptif ;
FORMAT IS "alpha(2)" ;

DEFINE ELEMENT modele-famille-dmc ;
DESCRIPTION  ;   codification  d'une  famille  de  véhicules-DMC,
                 faisant partie de l'identifiant du véhicule-DMC;
FORMAT IS "alpha(20)" ;

DEFINE ELEMENT mot-pour-bv ;
DESCRIPTION ;  codification du moteur à utiliser en relation avec
               une boîte de vitesses ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT mt-v-v-dpi ;
DESCRIPTION  ;  codification du moteur de la version  du véhicule
              DPI telle qu'elle apparait dans le descriptif ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT niveau-org-dmc ;
DESCRIPTION ;  niveau  de  raffinement  dans  la  définition d'un
                organe DMC.
                Contrainte : niveau-org-dmc =
                .   'famille-organe' si le code de la famille est
                différent de 'valeur inexistante'  et si le reste
                de   l'identifiant   de   l'organe   =   'valeur
                inexistante'
                .  'sous-famille-organe' si le code de la famille
                et le code de la sous-famille sont  différents de
                'valeur   inexistante'   et   si   le   reste  de
                l'identifiant de l'organe = 'valeur inexistante'
                . 'organe' si toutes les parties de l'identifiant
                de   l'organe   sont   différentes   de   'valeur
                inexistante'
                Ceci est vrai pour les moteurs et  les  boîtes de
                vitesses ;  ce n'est pas  vrai  pour  les châssis
                dont l'identifiant forme un tout indécomposable ;
FORMAT IS "alpha(20)" ;

DEFINE ELEMENT niveau-pie-dmc ;
DESCRIPTION ;  niveau de  raffinement  dans  la  définition d'une
                pièce DMC.
                Contrainte : niveau-pie-dmc =
                .    'famille-piece'  si  'code-fam-pie-dmc'  est
                différent   de   'valeur   inexistante'   et   si
                'code-type-pie-dmc'  et  'code-pie-dmc' = 'valeur
                inexistante'
```

```
                . 'type-piece' si 'code-fam-pie-dmc' et
                'code-type-pie-dmc' sont différents de 'valeur
                inexistante' et si 'code-pie-dmc' = 'valeur
                inexistante'
                . 'piece' si toutes les parties de l'identifiant
                de la pièce sont différentes de 'valeur
                inexistante' ;
FORMAT IS "alpha(20)" ;

DEFINE ELEMENT niveau-veh-dmc ;
DESCRIPTION ; niveau de raffinement dans la définition d'un
                véhicule DMC .
                Contrainte : niveau-veh-dmc =
                . 'gamme' si 'code-gamme-dmc' est différent de
                'valeur inexistante' et si le reste de
                l'identifiant = 'valeur inexistante'
                . 'famille' si 'code-gamme-dmc' et 'modele-
                famille-dmc' sont différents de 'valeur
                inexistante' et si le reste de l'identifiant =
                'valeur inexistante'
                . 'version' si toutes les parties de l'identi-
                fiant sont différentes de 'valeur inexistante' ;
FORMAT IS "alpha(10)" ;

DEFINE ELEMENT nom-ligne ;
DESCRIPTION ; nom identifiant d'une ligne de production ;
FORMAT IS "alpha(20)" ;

DEFINE ELEMENT nom-pays ;
DESCRIPTION ; nom identifiant du lieu de commercialiation de
               véhicules, d'organes ou de pièces;
FORMAT IS "alpha(15)" ;

DEFINE ELEMENT num-hypo ;
DESCRIPTION ; numéro d'hypothèse de nomenclature, de correspon-
              dance, de montage, etc... ;
FORMAT IS "alpha(10)" ;

DEFINE ELEMENT observations ;
DESCRIPTION ; observations (texte libre) quant à la présence ou
              non de la direction assistée, telle qu'elle
              apparait dans le descriptif ;
FORMAT IS "alpha(30)" ;

DEFINE ELEMENT option-mot ;
DESCRIPTION ; codification de l'option de moteur telle qu'elle
               apparait dans le code d'un moteur ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT quantité-composition ;
```

```
DESCRIPTION ;  quantité d'une pièce DMC composant une autre pièce
               DMC ;
FORMAT IS "num(3)" ;

DEFINE ELEMENT sous-famille-bv ;
DESCRIPTION  ;  codification de la sous-famille de boîte de
                vitesses telle qu'elle apparait dans le code
                d'une boîte de vitesses ;
FORMAT IS "alpha(1)" ;

DEFINE ELEMENT sous-famille-mot ;
DESCRIPTION ;  codification de la sous-famille de moteur telle
                qu'elle apparait dans le code d'un moteur ;
FORMAT IS "alpha(2)" ;

DEFINE ELEMENT version-codee-veh-dmc ;
DESCRIPTION  ;  codification de la version de véhicule DMC (en
               fonction de la motorisation par exemple), faisant
               partie de l'identifiant du véhicule DMC ;
FORMAT IS "alpha(5)" ;

DEFINE ELEMENT zone-geogr-veh-dmc ;
DESCRIPTION ; zone géographique de destination, de commercialisa-
              tion d'une version de véhicule DMC, faisant partie
              de l'identifiant du véhicule-DMC ;
FORMAT IS "alpha(15)" ;

DEFINE ELEMENT zone-pays ;
DESCRIPTION  ;  nom de la zone géographique à laquelle le pays
               appartient ;
FORMAT IS "alpha(20)" ;
```

ANNEXE 2 : LES ECRANS ORACLE

L'annexe 2 contient les descriptions des écrans ORACLE de la phase "Répartition-des-engagements" de l'application "Plans de pièces".

Ecran 1/1 :

BESOINS A ENGAGER PAR JOUR (CADENCE) :
0 0 0 0 0 5 2 3 3 4 2 4
LIGNES INACTIVES :

NOM :

CAPACITE : 0 0 0 0 0 0 0 0 0 0 0 0

ENGAGEMENTS : 0 0 0 0 0 0 0 0 0 0 0 0

NOM :

CAPACITE :

ENGAGEMENTS :

NOM :

CAPACITE :

ENGAGEMENTS :

==========

COMMENTAIRES :

ECRAN SUR LEQUEL SERONT AFFICHEES LES LIGNES INACTIVES .
ON REMARQUE QUE LES BESOINS A ENGAGER ONT ETE DIVISES PAR LE NOMBRE DE JOURS DE CHAQUE PERIODE POUR DONNER UNE CADENCE JOURNALIERE .

Ecran 1/2 :

BESOINS A ENGAGER PAR JOUR (CADENCE) :

	0	0	0	0	0	5	2	3	3	4	2	4

LIGNES INACTIVES :

NOM : LE_MANS_3

CAPACITE :	15	14	13	14	15	15	15	15	15	15	15	15

ENGAGEMENTS :

NOM : YOUGO

CAPACITE :	25	25	25	25	25	25	25	25	25	25	25	25

ENGAGEMENTS :

NOM :

CAPACITE :

ENGAGEMENTS :

==========

COMMENTAIRES :

L'UTILISATEUR, GRACE A LA TOUCHE "QUERY", VISUALISE LES LIGNES INACTIVES .

Ecran 1/3 :

```
BESOINS A ENGAGER PAR JOUR (CADENCE) :
              0    0    0    0    0    5    2    3    3    4    2    4
LIGNES INACTIVES :
-------------------------------------------------------------------------------
NOM :         LE_MANS_3

CAPACITE :    15   14   13   14   15   15   15   15   15   15   15   15

ENGAGEMENTS : 0    0    0    0    0    0005 0002 0003 0003 0004 0002 0004
-------------------------------------------------------------------------------
NOM :         YOUGO

CAPACITE :    25   25   25   25   25   25   25   25   25   25   25   25

ENGAGEMENTS :
-------------------------------------------------------------------------------
NOM :

CAPACITE :

ENGAGEMENTS :

===============================================================================

COMMENTAIRES :
--------------

L'UTILISATEUR DECIDE D'ENGAGER LES BESOINS JOURNALIERS SUR LA LIGNE "LE_MANS_3"
```

Ecran 2/1 :

BESOINS A ENGAGER PAR JOUR (CADENCE) :
0 0 0 0 0 5 2 3 3 4 2 4
LIGNES ACTIVES :

NOM :

CAPACITE :
ENGAG. EXISTANTS :
NOUVEAUX ENGAG. : 0 0 0 0 0 0 0 0 0 0 0 0

NOM :

CAPACITE :
ENGAG. EXISTANTS :
NOUVEAUX ENGAG. :

NOM :

CAPACITE :
ENGAG. EXISTANTS :
NOUVEAUX ENGAG. :

==========

COMMENTAIRES :

ECRAN QUI PERMETTRA A L'UTILISATEUR DE VISUALISER LES LIGNES ACTIVES .

Ecran 2/2 :

BESOINS A ENGAGER PAR JOUR (CADENCE) :
0 0 0 0 0 5 2 3 3 4 2 4
LIGNES ACTIVES :

NOM : LE_MANS_1

CAPACITE :	100	100	100	100	100	100	100	100	100	100	100	100
ENGAG. EXISTANTS :	90	90	90	90	90	90	90	90	90	90	90	90
NOUVEAUX ENGAG. :												

NOM :

CAPACITE :
ENGAG. EXISTANTS :
NOUVEAUX ENGAG. :

NOM :

CAPACITE :
ENGAG. EXISTANTS :
NOUVEAUX ENGAG. :

==========

COMMENTAIRES :

L'UTILISATEUR, GRACE A LA TOUCHE "QUERY", VISUALISE LES LIGNES ACTIVES POUVANT ENCORE ETRE ENGAGEES POUR LE MONTAGE D'UN MOTEUR .

Ecran 3/1 :

BESOINS A ENGAGER PAR JOUR (CADENCE) :

0 0 0 0 0 5 2 3 3 4 2 4

LIGNES A CREER :

NOM :

CAPACITE : 0 0 0 0 0 0 0 0 0 0 0 0

ENGAGEMENTS : 0 0 0 0 0 0 0 0 0 0 0 0

NOM :

CAPACITE :

ENGAGEMENTS :

NOM :

CAPACITE :

ENGAGEMENTS :

==========

COMMENTAIRES :

ECRAN QUI PERMETTRA A L'UTILISATEUR DE CREER DE NOUVELLES LIGNES .

Ecran 4/1 :

```
---------------------------------- PLAN PIECE ----------------------------------
! ORGANE : J6T HA      TYPE : M    PAYS : DAI       NUM-HYPO : HM               !
!-----------------------------------------------------------------------------!
! PERIODES : ! P1 ! P2 ! P3 ! P4 ! P5 ! P6 ! P7 ! P8 ! P9 ! P10! P11! P12      !
!------------+---+----+----+----+----+----+----+----+----+----+----+----------!
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! CAPACITES: !   !    !    !    !    !    !    !    !    !    !    !          !
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! ENGAGEMENT:!   !    !    !    !    !    !    !    !    !    !    !          !
!------------+---+----+----+----+----+----+----+----+----+----+----+----------!
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! CAPACITES: !   !    !    !    !    !    !    !    !    !    !    !          !
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! ENGAGEMENT:!   !    !    !    !    !    !    !    !    !    !    !          !
!------------+---+----+----+----+----+----+----+----+----+----+----+----------!
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! CAPACITES: !   !    !    !    !    !    !    !    !    !    !    !          !
!            !   !    !    !    !    !    !    !    !    !    !    !          !
! ENGAGEMENT:!   !    !    !    !    !    !    !    !    !    !    !          !
-------------------------------------------------------------------------------

===============================================================================

COMMENTAIRES :
--------------

ECRAN REPRESENTANT LE PLAN DE PIECES VIDE .
```

Ecran 4/2 :

PLAN PIECE

ORGANE : J6T HA TYPE : M PAYS : DAI NUM-HYPO : HM

PERIODES :	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
LE_MANS_3												
CAPACITES:	15	14	13	14	15	15	15	15	15	15	15	15
ENGAGEMENT:	0	0	0	0	0	5	2	3	3	4	2	4
CAPACITES:												
ENGAGEMENT:												
CAPACITES:												
ENGAGEMENT:												

COMMENTAIRES :

L'UTILISATEUR AYANT CHOISI D'ENGAGER TOUS LES BESOINS DU MOTEUR DUQUEL IL FAIT LE PLAN DE PIECES SUR LA LIGNE "LE_MANS_3", SEULE CETTE LIGNE EST VISUALISEE SUR L'ECRAN, LORSQUE L'UTILISATEUR EXECUTE UN "QUERY" .

ANNEXE 3 : IMPLEMENTATION DES FONCTIONS AVEC ORACLE

L'annexe 3 présente d'abord l'implémentation des fonctions de la phase "Répartition-des-engagements" avec le logiciel ORACLE.
Elle présente aussi l'implémentation avec ORACLE des aides à l'élaboration des hypothèses.

1. IMPLEMENTATION DES FONCTIONS

REMARQUES PRELIMINAIRES

1) Il est important d'avoir lu la description du processus de génération d'une maquette avec ORACLE (point 7.1.2.), afin de comprendre l'implémentation des fonctions que nous décrivons dans cette partie.
2) Nous décrivons l'implémentation des fonctions, non pas suivant la dynamique de celles-ci résultant de l'analyse fonctionnelle, mais suivant le regroupement de ces fonctions par bloc (cfr point 7.3.2.).
3) La description de l'implémentation est réduite. Nous parlons uniquement des principaux traitements et des principaux liens entre blocs, en laissant de côté les problèmes d'affichage, de positionnement des champs affichés sur les différentes pages d'écran, le nombre d'occurrences d'un record à afficher, etc...
4) Il est impossible de mettre, même en annexe, tout le code des fonctions implémentées. Aussi nous sommes-nous limités à mettre en annexe le code de quelques parties intéressantes.
5) Du point de vue de la forme, signalons que :
   * :nom-champ signifie : "contenu de nom-champ".
   * on peut préfixe un champ par le nom du bloc auquel il appartient. Exemple : engagei.pi-engage
   * ; en début de ligne signifie question posée par IAG lors du dialogue interactif de génération de l'application.

BLOC 0 : TRADUCTION

A partir des besoins de l'organe X pour le pays Y (pour lesquels l'utilisateur désire faire le plan de pièces), il faut traduire le besoin de chaque période en cadence journalière.

Exemple pour la période 1 :

Soient . le champ p1-besoins-m représentant le besoin
. les champs af1-p1-cadence,
af2-p2-cadence,

af3-p3-cadence, représentant les cadences qui doivent être affichées l'une sur l'écran de traitement des lignes inactives, l'aute sur l'écran de traitement des lignes actives et la dernière su l'écran de traitement des lignes créées.

La traduction s'opère par la requête SQL suivante, associée au champ p1-besoins-m :

```
; Field name :
p1-besoins-m
; ...
.
.
; SQL >
SELECT ROUND (:p1-besoins-m / 130)
INTO af1-p1-cadence, af2-p2-cadence, af3-p3-cadence
FROM DUMMY
```

Rappel : la table DUMMY permet de faire des calculs sur les données.

Les champs d'affichage, une fois remplis, permettent de gérer les liens entre les blocs.

BLOC 1 : ENGAGE1

1) Lorsque l'utilisateur visualise l'écran 1/1 (annexe 2), il faut qu'il puisse, par la touche-fonction QUERY, consulter les lignes inactives.
Réalisation : par une clause WHERE par défaut, qui sera automatiquement exécutée lorsque l'utilisateur fera le QUERY.

```
; Block name :
engage1
; Enter default WHERE and ORDER BY clause :
WHERE nom-ligne NOT IN (SELECT nom-ligne FROM montusi)
```

2) Pour choisir la ligne sur laquelle il désire travailler, l'utilisateur positionne simplement le curseur sur l'écran.

3) La création des engagements consiste en une introduction par l'utilisateur de ceux-ci.
Pour chaque valeur entrée, les contrôles suivants sont effectués: les contraintes besoin >= engagement
capacité >= engagement doivent être respectées.

Soient . le champ p1-engage contenant l'engagement introduit par l'utilisateur

. le champ afi-pi-cadence contenant la cadence requise (calculée, pour rappel, dans le bloc 0)
. le champ pi-ech contenant la capacité de la ligne.

Réalisation :

```
; Field name :
pi-engage
; ...
.
.
; SQL >
SELECT *
FROM DUMMY
WHERE (:engage1.pi-engage <= :traduction.afi-pi-cadence)
/
; Message if value not found :
Engagement > besoin
; Must value exist Y/N :
Y
SELECT *
FROM DUMMY
WHERE (:engage1.pi-engage <= :engage1.pi-ech)

; Message if value not found :
Engagement > capacité
; Must value exist Y/N :
Y
```

La combinaison de ces questions-réponses oblige l'utilisateur à ne rentrer que des engagements respectant les contraintes implémentées.

4) La création physique des engagements dans la table adéquate, "montusi", est effectuée par le TRIGGERS pré-UPDATE auquel on associe l'ordre SQL INSERT.

Réalisation :

```
; Field name :
PRE-UPDATE
; SQL >
INSERT INTO montusi VALUES (:traduction.id-m,
                            :traduction.type-m,
                            :engage1.nom-ligne,
                            :engage1.pi-engage,
                            :engage1.p2-engage,
                            ...
                            :engage1.pi4-engage)
```

Remarque : On peut considérer que le lien entre le bloc 0 et le bloc 1 est réalisé par l'affichage su le bloc 1 de champs dont les valeurs sont calculées dans le bloc 0 (les cadences).
Il est aussi réalisé lors du TRIGGERS, puisque l'insertion d'une ligne dans une table exige que la ligne soit complète. Il faut donc aller rechercher dans le bloc TRADUCTION les valeurs de id-m et type-m.

BLOC 2 : ENGAGE2

1) Comme dans le bloc 1, la consultation des lignes actives est réalisée par la clause WHERE par défaut.

2) Par rapport au bloc 1, un contrôle supplémentaire est implémenté, par une requête SQL : la contrainte "nouvel engagement + engagement existant <= capacité" doit être respectée pour chaque période.

3) La création physique des engagements dans la table "montusi" est effectuée par le TRIGGERS pré-UPDATE auquel on associe la requête SQL INSERT.

4) Le bloc 2 travaille sur la table "montusi". Or ce bloc a besoin des capacités de la ligne choisie afin d'effectuer les contrôles, lesquelles capacités se trouvent dans la table "ligne".
Afin de connaître les capacités de la ligne choisie, on utilise un TRIGGERS post-QUERY auquel on associe l'ordre SQL SELECT.

Soient . le champ p1-ech appartenant à la table "ligne" qui n'est pas celle référencée pa le bloc 2
. le champ p1-cap créé pour contenir, dans le bloc 2, la capacité recherchée
. engage2.nom-ligne contient le nom de la ligne sur laquelle on travaille dans le bloc 2.

Réalisation :

```
; Field name :
POST-QUERY
; SQL >
SELECT p1-ech INTO engage2.p1-cap
FROM ligne
WHERE nom-ligne = :engage2.nom-ligne
```

BLOC 3 : ENGAGE3

Les mêmes principes que ceux décrits pour les deux blocs précédents sont applicables ici.

BLOC 4 : CONTROLE

Le bloc 4 permet de faire le lien, au point de vue des valeurs de champs affichées, entre le bloc 3 et le bloc 5.

BLOC 5 : AFFICHAGE-PLAN

L'utilisateur, grâce à la touche-fonction "QUERY", peut visualiser le plan de pièces qu'il vient d'effectuer.
Le bloc 5 travaille sur la table "montusi". Lors du "QUERY", les valeurs des champs de cette table sont affichés, à savoir le nom de la ligne, les engagements, le nom de l'objet (organe ou pièce) et le type d'objet.
L'enregistrement de la table "ligne" affiché est celui pour lequel le nom de la ligne, le nom et le type de l'objet correspondent à ceux provenant du bloc de contrôle. Le lien est ainsi réalisé.

Exemple : pour le champ id-m, on a

```
; Field name :
ID-M
; ...
.
.
; Is this field in the base table Y/N :
Y
; Is this field part of the primary key Y/N :
Y
; Field to copy primary key from :
CONTROLE. ID-M
; ...
```

On est donc sûr que les lignes affichées sont celles pour lesquelles l'utilisateur a donné des engagements dans le cadre de la réalisation du plan de pièces en question.
De plus, on affiche, pour chaque ligne, ses capacités, lesquelles ne se trouvent pas dans la table "montusi". On retrouve les capacités p1-ech, p2-ech, ...,p14-ech dans la table "ligne" par un TRIGGERS post-QUERY et les valeurs sont retournées dans les champs p1-cap, p2-cap, ..., p14-cap appartenant au bloc AFFICHAGE-PLAN.

```
; Field name :
```

```
POST-QUERY
; SQL >
SELECT p1-ech
INTO affichage-plan.p1-cap
FROM ligne
WHERE nom-ligne = :affichage-plan.nom-ligne
```

2. L'AIDE A L'ELABORATION DES HYPOTHESES

Nous avons parlé au point 2.3.3. des aides passives à l'élaboration d'hypothèses, qui interviennent sous la forme de sélections dans la base de données de solutions possibles aux problèmes que se pose l'utilisateur.

Les requêtes peuvent se situer à trois niveaux. Donnons-en des exemples d'implémentation avec ORACLE. Ces exemples n'ont pas nécessairement été implémentés explicitement, sous la forme présentée dans l'application ci-dessus.

A. Requêtes passives de niveau 1

Rappel : les requêtes passives de niveau 1 font intervenir, lors de la sélection, des critères simples.

Exemple : les lignes inactives sont celles appartenant à la table "ligne" et n'appartenant pas à la table "montusi".
Il s'agit d'une clause WHERE par défaut.

```
[ SELECT nom-ligne
  FROM ligne ]
WHERE nom-ligne NOT IN ( SELECT nom-ligne
                         FROM montusi )
```

B. Requêtes passives de niveau 2

Rappel : les requêtes passives de niveau 2 font intervenir des critères liés à l'objet de la sélection.

Exemple : on veut sélectionner les lignes de production qui montent des moteurs, ainsi que leurs capacités.

```
SELECT nom-ligne, p1-ech, p2-ech, ..., p14-ech
FROM ligne
WHERE nom-ligne IN ( SELECT nom-ligne
                     FROM montusi
                     WHERE type-m = 'M' )
```

C. Requêtes passives de niveau 3

Rappel : les requêtes passives de niveau 3 font intervenir des critères croisés.

Exemple : on veut sélectionner les lignes de production (et leurs capacités) qui montent des moteurs et dont, pour chaque période, l'engagement est inférieur au besoin du moteur 'A1M' pour le pays 'Belgique'. Il s'agit de la sélection des lignes actives pouvant encore monter.

```
SELECT nom-ligne, p1-ech, p2-ech, ..., p14-ech
FROM ligne
WHERE nom-ligne IN ( SELECT nom-ligne
                     FROM montusi
                     WHERE type-m = 'M'
                       AND p1-engage < ( SELECT p1-besoins-m
                                         FROM comm-m
                                         WHERE id-m = 'A1M'
                                           AND nom-pays =
                                               'Belgique' )
                       AND ...
                       AND p14-engage < ( SELECT p14-besoins-m
                                          FROM comm-m
                                          WHERE id-m = 'A1M'
                                            AND nom-pays =
                                                'Belgique' )
```

D. Autres types de requêtes

Nous donnons deux autres exemples d'implémentation de requêtes avec ORACLE.

1) Le calcul des cadences

```
SELECT ROUND (:p1-besoins-m / 130)
INTO af1-p1-cadence
FROM DUMMY
```

(la requête a été implémentée dans le bloc 0)

2) Le contrôle de validité des engagements

```
; SQL >
SELECT *
FROM DUMMY
WHERE (p1-engage <= af1-p1-cadence)
; Message if value not found :
Engagement > besoin
; Must value exist Y/N :
Y
```

(la requête a été implémentée dans les blocs d'engagement)

ANNEXE 4 : IMPLEMENTATION DES FONCTIONS AVEC DSL-PROTO

L'annexe 4 présente d'abord l'implémentation des fonctions de la phase "Répartition-des-engagements" avec le logiciel DSL-PROTO. Elle présente aussi l'implémentation avec DSL-PROTO des aides à l'élaboration des hypothèses.

REMARQUES PRELIMINAIRES

1) La description de l'implémentation est réduite. Nous parlons uniquement des principaux traitements réalisés par DSL-PROTO pour implémenter les fonctions de la phase "Répartion-des-engagements". Nous laissons de côté les problèmes d'affichage, de transit des messages, etc...

2) Pour chaque fonction, nous donnons un algorithme en pseudo-langage, afin de fixer la démarche de traitement. Nous ajoutons éventuellement quelques commentaires sur les fonctionnalités de DSL-PROTO.

3) Les explications concernant les algorithmes sont progressives. Par conséquent, la compréhension de certaines instructions de la fonction 4 prérequièrent l'étude des fonctions 1, 2, 3.

Fonction 1 : sélection-lignes-actives

```
FOR EACH ligne-product
    FOR EACH montage-boite-v
        WHERE nom-ligne OF montage-boite-v == nom-ligne OF
              ligne-product
        WITH COUNTER compteur
        ** on a trouvé une ligne active.
        [ Calcul du total de ses engagements pour chaque
        période. ]
    ENDFOR montage-boite-v
    FOR EACH montage-chassis...¦
    FOR EACH montage-moteur....¦ même traitement que pour boite-v
    FOR EACH usinage...........¦
    IF compteur == 0
    THEN
        ** la ligne est inactive et on passe à la ligne suivante
        NEXT BLOCK-FOR ligne-product
    ELSE
        [ Vérification que, pour chaque période, les engagements
          existants sont inférieurs aux capacités. ]
        IF oK
        THEN
            ** on peut sélectionner la ligne
            [ Génération d'un message contenant le nom de la
              ligne. ]
        ENDIF
    ENDIF
ENDFOR ligne-product
```

Commentaires :

Le principe consiste à, pour chaque occurrence de ligne de production, et pour chaque occurrence de montage ou d'usinage qu'elle possède, sommer les engagements. Un compteur (instruction WITH COUNTER) permet de connaître le nombre d'occurrences de montage ou d'usinage trouvées. Si ce compteur vaut 0 en sortie des boucles portant sur le montage ou l'usinage, la ligne est inactive et on passe à l'occurrence suivante de ligne-product. Sinon, on vérifie que la ligne est encore capable de monter ou d'usiner (c'est-à-dire que ses engagements sont inférieurs à sa capacité).

Fonction 2 : sélection-lignes-inactives

```
FOR EACH ligne-product
    FOR EACH montage-boite-v
        WHERE nom-ligne OF ligne-product == nom-ligne OF montage-
              boite-v
        ** on sort car la ligne trouvée est active
        NEXT BLOCK-FOR ligne-product
    ENDFOR montage-boite-v
    FOR EACH montage-chassis...¦
    FOR EACH montage-moteur....¦ même traitement que pour boite-v
    FOR EACH usinage...........¦
    **  on  n'est pas sorti :  la  ligne  est  inactive  :  on la
        sélectionne
    [ Génération d'un message contenant le nom de la ligne ]
ENDFOR ligne-product
```

Commentaires :

Le principe est de regarder, pour chaque occurrence de ligne-product si elle possède une ou plusieurs occurrences de montage ou d'usinage, auquel cas on peut arrêter de travailler sur l'occurrence référencée, puisque la ligne est active.
La sortie prématurée de bloc (instruction NEXT BLOCK-FOR) permet de n'arriver au bout du bloc ligne-product qu'avec une ligne inactive.

Fonction 3 : traduction-cadence

```
IF <le plan de pièces à faire concerne une boîte de vitesses>
THEN
    FOR EACH commerce-bv-dmc
        WHERE (  id-bv-dmc OF commerce-bv-dmc ==  <nom d'objet du
                 plan de pièces à faire> )
          AND  (  nom-pays OF commerce-bv-dmc ==  <nom de pays du
                 plan de pièces à faire> )
        **  on a l'occurrence de la relation voulue. On peut
            générer un message contenant les cadences, qui
            correspondent aux besoins contenus dans l'occurrence
            référencée divisés par le nombre de jours.
        [ Calcul des cadences ]
        [ Génération d'un message contenant les cadences ]
    ENDFOR commerce-bv-dmc
ELSE
    idem pour commerce-cha-dmc, commerce-mot-dmc, commerce-pie-
    dmc.
ENDIF
```

Commentaires :

- Une seule occurrence de commerce-xx-dmc sera trouvée, puisque la clause WHERE porte sur les 2 identifiants de la relation.
- Le calcul des cadences s'effectue de la manière suivante :
on boucle sur les périodes (i) [boucle WHILE]

```
LET cadence [ i ] := TRUNCATE ( besoins-dyn-bv [ i ] OF commerce-
                                bv-dmc : 130 )
```

Fonction 4 : choix-lignes-possibles

```
[ Réception des occurrences de message contenant les lignes
   choisies par   l'utilisateur ]
**  on transfère le contenu des occurrences de message dans un
    message indicé de travail, l'indice permettant de déterminer
    qu'il y a compteur-1 occurrences de message.
[ Transfert du contenu des occurrences de message ]
```

Même traitement pour les lignes actives et inactives sélectionnées auparavant (compteur-2).

```
i := 1
WHILE i <= compteur-1
      trouvé := F
      j := 1
      WHILE j <= compteur-2 AND trouvé == F
            IF nom-ligne [ i ] == nom-ligne [ j ]
            THEN
                ** la ligne choisie existe dans celles proposées
                [ Génération d'un message contenant le nom de la
                  ligne valide ]
            ELSE
                j := j + 1
            ENDIF
      ENDWHILE
      IF trouvé == F
      THEN
          ** la ligne n'existe pas dans celles proposées
          [ Génération d'un message d'erreur ]
      ENDIF
      i := i + 1
ENDWHILE
```

Commentaires :

- Principe de l'algorithme :
étant donné 2 tableaux, le premier contenant les lignes choisies, le second les lignes possibles, on balaie le second tableau pour chaque élément du premier, jusqu'à ce qu'on trouve l'élément correspondant.
Si on arrive au bout du deuxième tableau sans "matching", la ligne choisie n'est pas valable.

- Ce développement est nécessaire vu que DSL-PROTO ne peut travailler que sur une occurrence de message à la fois.
Il s'agit de l'instruction

```
RECEIVES EACH <nom-message>
```

```
    instructions
ENDRECEIVES <nom-message>
```

où les instructions sont exécutées autant de fois qu'il y a d'occurrences du message reçues.
Le transfert dans le message indicé de travail (w-lignes-choisies) s'effectue comme suit :

```
LET compteur-1 := 1
RECEIVES EACH lignes-choisies-utilisateur
LET nom-ligne [ compteur-1 ] OF w-lignes-choisies :=
    nom-ligne OF lignes-choisies-utilisateur
LET compteur-1 := compteur-1 + 1
ENDRECEIVES lignes-choisies-utilisateur
LET compteur-1 := compteur-1 - 1
```

- C'est dans la fonction 3 qu'a été demandé à l'utilisateur de choisir certaines des lignes proposées par la génération de messages.

Fonction 5 : création-nouvelles-lignes

```
RECEIVES EACH lignes-créées
OPENADDS ligne-product
**  transfert du contenu de l'occurrence du message lignes-créées
    référencée dans les éléments de ligne-product
ADDS ligne-product
ENDRECEIVES lignes-créées
```

Commentaires :

Cet algorithme simple met en évidence trois principes
- l'algorithme est effectué pour chaque ligne créée par l'utilisateur
- on ne peut accéder au contenu d'une entité ou d'une relation à ajouter qu'à l'intérieur d'un bloc de création (OPENADDS ...ADDS)
- les structures de bloc permettent au concepteur de délimiter les différents types de traitements à effectuer.

Fonction 6 : mise-à-jour-engagements

```
IF <le plan de pièces à faire concerne une boîte de vitesses>
THEN
    FOR EACH montage-boite-v
        WHERE ( nom-ligne OF montage-boite-v == nom-ligne OF
                lignes-choisies-i )
          AND ( id-bv-dmc OF montage-boite-v == <nom d'objet du
                plan de pièces à faire> )
        ** on peut afficher les engagements et générer un message
           les contenant
        [ Affichage des engagements ]
        [ Génération d'un message contenant les engagements ]
    ENDFOR montage-boite-v
ELSE
    idem pour montage-chassis, montage-moteur, usinage
ENDIF
```

Commentaires :

Etant donné que
- il y a des modifications dues au dialogue (cfr point 9.2.3.)
- toute mise-à-jour des engagements n'est physiquement effectuée qu'après le choix de validation par l'utilisateur, puisque, par souci de simplification, le numéro d'hypothèse n'est pas géré ici (contrairement à l'implémentation de la phase avec ORACLE)
- une mise-à-jour immédiate dans la base de données, sans gestion du numéro d'hypothèse, poserait des problèmes en cas de non validation du plan de pièces à faire par l'utilisateur,

la fonction se limite à afficher les engagements existants et d'autres informations (fonction DISPLAY) et à générer un message les contenant.

Fonction 7 : création-engagements

Pour les mêmes raisons que la fonction 6, le travail se limite ici à afficher des informations guidant l'utilisateur dans la création des engagements pour les lignes inactives ou créées choisies.

Fonction 8 : contrôle-validité-engagements

```
** premier contrôle de validité : engagements < capacités
FOR EACH ligne-product
    WHERE   nom-ligne   OF   ligne-product    ==   nom-ligne   OF
            lignes-choisies-2
    LET i := 1
    LET erreur := F
    WHILE ( i <= 14 ) AND ( NOT erreur )
          IF <la ligne choisie est active>
          THEN
              IF <l'  engagement existant [ i ] + le nouvel enga-
                  gement [ i ] est supérieur à la capacité  [ i ]
                  de la ligne>
              THEN
                  LET erreur := T
              ENDIF
              LET i = i + 1
          ELSE
              IF <le nouvel engagement [ i ] est  supérieur  à la
                  capacité [ i ] de la ligne>
              THEN
                  LET erreur := T
              ENDIF
              LET i = i + 1
          ENDIF
    ENDWHILE
ENDFOR ligne-product
** deuxième contrôle de validité : engagements < besoins
LET i := 1
LET erreur-1 := F
WHILE ( i <= 14 ) AND ( NOT erreur-1 )
      IF <l'engagement [ i ] est supérieur au besoin [ i ]>
      THEN
          LET erreur-1 := T
      ENDIF
      LET i := i + 1
ENDWHILE
** vérification de la présence ou non d'erreurs
IF erreur OR erreur-1
THEN
    [ Affichage message d'erreur ]
    [ Génération  d'un message contenant  le nom de la  ligne non
      valide ]
ELSE
    [ Génération d'un message contenant le nom de la ligne valide
      à destination du point de synchonisation ]
ENDIF
```

Commentaires :

Le principe de l'algorithme est le suivant :
étant donné une ligne choisie par l'utilisateur dont les engagements viennent d'être soit mis-à-jour, soit créés, il faut vérifier la validité de ces engagements.

Premier contrôle : il faut que les engagements soient inférieurs aux capacités. On référence donc dans la base de données l'occurrence de ligne-product correspondant à la ligne choisie sur laquelle on travaille, on vérifie la contrainte pour chaque période (boucle WHILE), en sortant de la boucle dès qu'on a trouvé une erreur. Le contrôle est différent suivant qu'il y a ou non des engagements existants.

Second contrôle : il faut que les engagements soient inférieurs aux besoins. Les engagements (mis-à-jour ou créés), ainsi que les besoins, sont contenus dans des messages. Le contrôle est donc effectué par une boucle sur les périodes, dont on sort dès que la contrainte n'est plus respectée.

Au terme de ces deux contrôles, si l'un de ceux-ci a permis de détecter une erreur, on affiche un message d'erreur et on génère un message contenant le nom de la ligne pour laquelle il faut mettre à jour les engagements erronnés. Sinon, on génère un message contenant le nom de la ligne, à destination du point de synchonisation.

Une fonctionnalité importante de DSL-PROTO est mise en évidence dans la fonction 8 : la génération d'un message permettant la contribution à la réalisation du point de synchronisation, au cas où il n' y a pas d'erreur et la génération d'un message permettant de corriger les engagements (en activant à nouveau la fonction 6 (mise-à-jour-engagements) s'il y a erreur.
On illutre donc bien la dynamique par les messages.

Fonction 9 : contrôle-total-engagements

```
[ Réception de chaque ligne et de ses engagements ; transfert de
  ceux-ci dans un message indicé de travail. Il y en a
  "compteur" ]
[ Réception des besoins à engager ]
LET i := 1
LET erreur := F
WHILE ( i <= compteur ) AND ( NOT erreur )
      LET j := 1
      WHILE ( j <= 14 ) AND ( NOT erreur )
            [ ajout au total des engagements [ j ] de
              l'engagement [ j ] ]
            IF <le total des engagements [ j ] est supérieur au
                besoin [ j ]>
            THEN
                LET erreur := T
            ENDIF
            LET j := j + 1
      ENDWHILE
      LET i := i + 1
ENDWHILE
IF erreur
THEN
    [ Affichage message d'erreur ]
ELSE
    [ Affichage du plan de pièces établi ]
ENDIF
```

Commentaires :

Principe de l'algorithme :
- on met les engagements et les noms des lignes dans des messages indicés de travail, pour toutes les lignes choisies par l'utilisateur et valide ligne par ligne
- on met les besoins dans un message de travail
- ensuite, par deux boucles WHILE imbriquées, la première portant sur les lignes, la seconde sur les périodes, on somme les engagements pour chaque période, en évitant le travail inutile puisqu'il y a sortie de boucle dès que le total des engagements dépasse le besoin, pour une période
- ensuite, s'il y a erreur, on prévient l'utilisateur qu'il n'y a pas de validation possible du plan de pièces. Sinon, le plan de pièces peut être validé et on affiche le plan de pièces établi par l'utilisateur.

Les aspects de cet algorithme propres à DSL-PROTO ont déjà été développés dans les fonctions précédentes.

Fonction 10 : validation-plan-pièces

```
[ Réception du choix de l'utilisateur ]
IF <le choix est de valider le plan de pièces établi>
THEN
    [ Réception de chaque ligne et de ses engagements et
      transfert dans un message indicé de travail ]
    LET i := 1
    WHILE i <= compteur
          IF <le plan de pièces concerne une boîte de vitesses>
          THEN
              IF <la ligne référencée par l'indice i est active>
              THEN
                  FOR EACH montage-boite-v
                      WHERE ( id-bv-dmc OF montage-boite-v ==
                              <nom d'objet du plan de pièces à
                              faire> )
                        AND ( nom-ligne OF montage-boite-v ==
                              <nom de la ligne du message indicé
                              de travail> )
                      BEGINMODIFIES montage-boite-v
                      [ Mise-à-jour des éléments de montage-
                        boite-v ]
                      MODIFIES montage-boite-v
                  ENDFOR montage-boite-v
              ELSE
                  OPENADDS montage-boite-v
                  [ Remplissage des éléments de la relation ]
                  ADDS montage-boite-v
              ENDIF
          ELSE
              idem pour montage-chassis, montage-moteur, usinage
          ENDIF
          LET i := i + 1
    ENDWHILE
ELSE
    ** l'utilisateur ne désire pas valider le plan de pièces
    [ Affichage d'informations complémentaires ]
ENDIF
```

Commentaires :

Le principe de l'algorithme est de recevoir le message contenant le choix de l'utilisateur de valider ou non le plan de pièces établi. Si l'utilisateur veut valider, après avoir mis le contenu des occurrences de messages reçues dans un message indicé de travail, on effectue, pour chaque indice du message de travail (boucle i) soit la mise-à-jour des engagements (bloc BEGINMODI FIES... MODIFIES) soit la création des engagements (bloc OPENADDS

... ADDS), en fonction de l'état de la ligne : active ou inactive/créée.

Mettons en évidence ce que nous avons déjà constaté à plusieurs reprises dans les algorithmes des différentes fonctions :
il n'y a pas de possibilité de déclaration et d'appel de procédures avec paramètres.
Ceci nous oblige, lorsque le même traitement doit être effectué sur des entités ou des relations différentes (exemple : montage-boite-v, montage-chassis, montage-moteur, usinage) en fonction d'un paramètre (exemple : le type d'objet sur lequel le plan de pièces est à faire), à travailler avec des structures conditionnelles imbriquées dans lesquelles on répète le même algorithme portant sur une entité ou une relation différente.
Il serait intéressant d'avoir la possibilité de déclarer la procédure

```
modification ( <nom-relation> )
...
FOR EACH <nom-relation>
...
BEGINMODIFIES <nom-relation>
...
```

Les structures conditionnelles imbriquées seraient réduites à :

```
IF <le plan de pièces est à faire pour une boîte de vitesses>
THEN
    CALL modification ( montage-boite-v )
ELSE
    IF <le plan de pièces est à faire pour un châssis>
    THEN
        CALL modification ( montage-chassis )
...
```

Remarque : on trouve en annexe 6 le code complet de la fonction validation-plan-pièces.

L'AIDE A L'ELABORATION DES HYPOTHESES

Nous avons parlé des aides passives à l'élaboration d'hypothèses, qui interviennent sous la forme de sélections dans la base de données de solutions possibles aux problèmes que se pose l'utilisateur.

Les requêtes peuvent se situer à trois niveaux. Comme lors de la description des fonctions réalisées avec ORACLE, donnons-en des exemples d'implémentation avec DSL-PROTO. Ces exemples n'ont pas nécessairement été implémentés explicitement ; ce sont ceux présentés au point 7.4.

A. Requêtes passives de niveau 1

Rappel : les requêtes passives de niveau 1 font intervenir, lors de la sélection, des critères simples.

Exemple : les lignes inactives sont celles qui n'ont aucune relation de montage ou d'usinage.
Il s'agit ici d'une boucle FOR EACH sur ligne-product, contenant 4 boucles FOR EACH pour les quatre relations.
On sort de la boucle ligne-product dès qu'on trouve une ligne de production ayant une relation. Si on ne sort pas prématurément de cette boucle, on a trouvé une ligne inactive.

```
FOR EACH ligne-product
    FOR EACH montage-boite-v
        WHERE nom-ligne OF montage-boite-v == nom-ligne OF
              ligne-product
        NEXT BLOCK-FOR ligne-product
    ENDFOR montage-boite-v
    FOR EACH montage-chassis ... ¦
    FOR EACH montage-moteur .... ¦ même traitement que pour la
    FOR EACH usinage ........... ¦ boîte de vitesses
ENDFOR ligne-product
```

B. Requêtes passives de niveau 2

Rappel : les requêtes passives de niveau 2 font intervenir des critères liés à l'objet de la sélection.

Exemple : on veut sélectionner les lignes de production qui montent des moteurs, ainsi que leurs capacités.

```
FOR EACH ligne-product
    FOR EACH montage-moteur
```

```
        WHERE nom-ligne OF ligne-product == nom-ligne OF montage-
              moteur
        [ Traitement du  nom de  la ligne sélectionnée et  de ses
          capacités ]
        NEXT BLOCK-FOR ligne-product
    ENDFOR montage-moteur
ENDFOR ligne-product
```

C. Requêtes passives de niveau 3

Rappel : les requêtes passives de niveau 3 font intervenir des critères croisés.

Exemple : on veut sélectionner les lignes de production (et leurs capacités) qui montent des moteurs et dont, pour chaque période, l'engagement est inférieur au besoin du moteur 'A1M' pour le pays 'Belgique'. Il s'agit de la sélection des lignes actives pouvant encore monter.

```
FOR EACH ligne-product
    FOR EACH montage-moteur
        WHERE nom-ligne OF ligne-product == nom-ligne OF montage-
              moteur
        FOR EACH commerce-mot-dmc
            WHERE nom-pays == "BELGIQUE" AND id-mot-dmc == "A1M"
            LET erreur := F
            LET i := 1
            WHILE ( NOT erreur ) AND ( i <= 14 )
                  IF engage [ i ] OF montage-moteur >
                     besoins-dyn-mot [ i ] OF commerce-mot-dmc
                  THEN
                      LET erreur := T
                  ENDIF
                  LET i := i + 1
            ENDWHILE
            IF erreur == F
            THEN
                [ Traitement de la ligne sélectionnée ]
            ENDIF
            NEXT BLOCK-FOR ligne-product
        ENDFOR commerce-mot-dmc
    ENDFOR montage-moteur
ENDFOR ligne-product
```

D. Autres types de requêtes

Les autres exemples d'implémentation de requêtes avec DSL-PROTO, tels que le calcul des cadences et le contrôle de validité des engagements, sont évidents de par les instructions possibles dans le langage PROTO.

ANNEXE 5 : CODE DU BLOC D'AFFICHAGE AVEC ORACLE

L'annexe 5 présente le code d'une partie du bloc d'affichage du plan de pièces avec le logiciel ORACLE.

```
;Block name / Description :
AFFPLAN
;Table name :
MONTUSI
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
3
;Base crt line ?
6
;How many physical lines per record ?
5
;Field name :
NOM_LIGNE
;Type of field :
CHAR
;Length of field / Display length / Query length :
20/11
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
;Default value :



;Page :
6
;Line :
1
;Column :
2
;Prompt :
;Allow field to be entered Y/N :
Y
;SQL>
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
;Lowest value :
;Highest value :
```

```
;Field name :
ID_M
;Type of field :
CHAR
;Length of field / Display length / Query length :
20/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
CONTR.ID_M
;Page :

;SQL>

;Field name :
TYPE_M
;Type of field :
CHAR
;Length of field / Display length / Query length :
1
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
CONTR.TYPE_M
;Page :

;SQL>
```

```
;Field name :
NO_HYPO
;Type of field :
CHAR



;Length of field / Display length / Query length :
10
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
CONTR.NO_HYPO
;Page :

;SQL>

;Field name :
P1_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
15
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P2_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
20
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P3_CAP
;Type of field :
NUMBER



;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
25
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P4_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
30
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P5_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
35
;Prompt :

;Allow field to be entered Y/N :
N



;SQL>

;Field name :
P6_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
40
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P7_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
45
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P8_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2


;Column :
50
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P9_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
55
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P10_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
60
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P11_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N


;Default value :

;Page :
6
;Line :
2
;Column :
65
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P12_CAP
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
70
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P1_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
4
;Column :
15
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P2_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
20
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P3_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
25
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P4_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :



;Page :
6
;Line :
4
;Column :
30
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P5_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
35
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

Field name :
P6_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
40
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>



;Field name :
P7_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
45
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P8_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
50
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P9_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :



;Page :
6
;Line :
4
;Column :
55
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
P10_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
60
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>
```

```
;Field name :
P11_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
65
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>



;Field name :
P12_ENGAGE
;Type of field :
NUMBER
;Length of field / Display length / Query length :
4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
70
;Prompt :

;Allow field to be entered Y/N :
N
;SQL>

;Field name :
*POST-QUERY
;SQL>
SELECT P1_ECH
INTO APPPLAN.P1_CAP
FROM LIGNE
WHERE NOM_LIGNE = :APPPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P2_ECH
INTO APPPLAN.P2_CAP
FROM LIGNE
WHERE NOM_LIGNE = :APPPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P3_ECH
INTO APPPLAN.P3_CAP
FROM LIGNE
WHERE NOM_LIGNE = :APPPLAN.NOM_LIGNE
/
;Message if value not found :
```

```
;Must value exist Y/N :
N
SELECT P4_ECH
INTO AFFPLAN.P4_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :



;Must value exist Y/N :
N
SELECT P5_ECH
INTO AFFPLAN.P5_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P6_ECH
INTO AFFPLAN.P6_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P7_ECH
INTO AFFPLAN.P7_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P8_ECH
INTO AFFPLAN.P8_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P9_ECH
INTO AFFPLAN.P9_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P10_ECH
INTO AFFPLAN.P10_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :
```

```
;Must value exist Y/N :
N
SELECT P11_ECH
INTO AFFPLAN.P11_CAP
FROM LIGNE


WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE
/
;Message if value not found :

;Must value exist Y/N :
N
SELECT P12_ECH
INTO AFFPLAN.P12_CAP
FROM LIGNE
WHERE NOM_LIGNE = :AFFPLAN.NOM_LIGNE

;Message if value not found :

;Must value exist Y/N :
N
;Field name :
```

ANNEXE 6 : CODE D'UNE FONCTION AVEC DSL-PROTO

L'annexe 6 contient le code de la fonction "validation-plan -pièce" avec le logiciel DSL-PROTO.

```
DEFINE PROCESS validation-plan-piece :

RECEIVES ensemble-ligne-1,
         plan-piece-7-a-faire,
         choix-validation :
GENERATES w-engagement-utilisateur;
ADDS usinage;
ADDS montage-boite-v;
ADDS montage-chassis;
ADDS montage-moteur ;
MODIFIES usinage;
MODIFIES montage-boite-v;
MODIFIES montage-chassis;
MODIFIES montage-moteur ;
TRIGGERED BY GENERATION OF plan-piece-7-a-faire :
PROTOTYPING-PROCEDURE :
PROCEDURE validation
DEFINE compteur FORMAT IS "INT(2)"
DEFINE i        FORMAT IS "INT(2)"
DEFINE j        FORMAT IS "INT(2)"
OPENGENERATES w-engagement-utilisateur
RECEIVES choix-validation
RECEIVES plan-piece-7-a-faire
IF ( choix-v OF choix-validation == "O" )
THEN
    LET compteur := 1
    RECEIVES EACH ensemble-ligne-1
    LET nom-ligne OF w-g-engagement-utilisateur [ compteur ] OF
        w-engagement-utilisateur := nom-ligne OF ensemble-ligne-1
    LET l-a OF w-g-engagement-utilisateur [ compteur ] OF
        w-engagement-utilisateur := l-a OF ensemble-ligne-1
    LET i := 1
    WHILE i <= 6
          LET engage-ligne [ i ] OF w-g-engagement-utilisateur [ compteur ]
              w-engagement-utilisateur := engage-ligne [ i ] OF
              ensemble-ligne-1
          LET engage-utilisateur [ i ] OF w-g-engagement-utilisateur
              [ compteur ] OF
              w-engagement-utilisateur := engage-utilisateur [ i ] OF
              ensemble-ligne-1
          LET i := i + 1
    ENDWHILE
    LET compteur := compteur + 1
    ENDRECEIVES ensemble-ligne-1
    LET i := 1
    WHILE i <= compteur - 1
          IF code-objet-m OF plan-piece-7-a-faire == "B"
          THEN
              IF l-a OF w-g-engagement-utilisateur [ i ]
                 OF w-engagement-utilisateur == "O"
              THEN
                  FOR EACH montage-boite-v
                      WHERE ( id-bv-dmc OF montage-boite-v ==
                              nom-objet-m OF plan-piece-7-a-faire ) .AND.
                            ( nom-ligne OF montage-boite-v ==
                              nom-ligne OF w-g-engagement-utilisateur [ i ]
                              w-engagement-utilisateur )
                      BEGINMODIFIES montage-boite-v
```

```
            LET i := 1
            WHILE i <= 6
                LET engagements-bv [ i ] OF montage-boite-v :=
                    engagements-bv [ i ] OF montage-boite-v +
                    engage-utilisateur [ j ] OF
                    w-g-engagement-utilisateur [ i ] OF
                    w-engagement-utilisateur
                LET i := i + 1
            ENDWHILE
            MODIFIES montage-boite-v
        ENDFOR montage-boite-v
    ELSE
        OPENADDS montage-boite-v
        LET id-bv-dmc OF montage-boite-v := nom-objet-m OF
        plan-piece-7-a-faire
        LET nom-ligne OF montage-boite-v := nom-ligne OF
        w-g-engagement-utilisateur [ i ] OF
        w-engagement-utilisateur
        LET i := 1
        WHILE i <= 6
            LET engagements-bv [ i ] OF montage-boite-v :=
                engage-utilisateur [ i ] OF
                w-g-engagement-utilisateur [ i ] OF
                w-engagement-utilisateur
            LET i := i + 1
        ENDWHILE
        ADDS montage-boite-v
    ENDIF
ELSE
    IF code-objet-m OF plan-piece-7-a-faire == "C"
    THEN
        IF 1-a OF w-g-engagement-utilisateur [ i ]
           OF w-engagement-utilisateur == "O"
        THEN
            FOR EACH montage-chassis
              WHERE ( id-cha-dmc OF montage-chassis ==
                      nom-objet-m OF plan-piece-7-a-faire ) .AND.
                    ( nom-ligne OF montage-chassis ==
                      nom-ligne OF w-g-engagement-utilisateur [ i ]
                        w-engagement-utilisateur )
                BEGINMODIFIES montage-chassis
                LET i := 1
                WHILE i <= 6
                    LET engagements-cha [ i ] OF montage-chassis
                        engagements-cha [ i ] OF montage-chassis
                        engage-utilisateur [ i ] OF
                        w-g-engagement-utilisateur [ i ] OF
                        w-engagement-utilisateur
                    LET i := i + 1
                ENDWHILE
                MODIFIES montage-chassis
            ENDFOR montage-chassis
        ELSE
            OPENADDS montage-chassis
            LET id-cha-dmc OF montage-chassis := nom-objet-m OF
            plan-piece-7-a-faire
            LET nom-ligne OF montage-chassis := nom-ligne OF
            w-g-engagement-utilisateur [ i ] OF
            w-engagement-utilisateur
            LET i := 1
```

```
            WHILE j <= 6
                  LET engagements-cha [ i ] OF montage-chassis :=
                      engage-utilisateur [ j ] OF
                      w-g-engagement-utilisateur [ i ] OF
                      w-engagement-utilisateur
                  LET j := j + 1
            ENDWHILE
            ADDS montage-chassis
        ENDIF
    ELSE
        IF code-objet-m OF plan-piece-7-a-faire == "M"
        THEN
            IF l-a OF w-g-engagement-utilisateur [ i ]
               OF w-engagement-utilisateur == "O"
            THEN
                FOR EACH montage-moteur
                 WHERE ( id-mot-dmc OF montage-moteur ==
                         nom-objet-m OF plan-piece-7-a-faire ) .AND.
                       ( nom-ligne OF montage-moteur ==
                         nom-ligne OF w-g-engagement-utilisateur [ i
                           OF w-engagement-utilisateur )
                    BEGINMODIFIES montage-moteur
                    LET j := 1
                    WHILE j <= 6
                          LET engagements-mot [ j ] OF montage-mote
                              :=
                              engagements-mot [ j ] OF montage-mote
                              +
                              engage-utilisateur [ i ] OF
                              w-g-engagement-utilisateur [ i ] OF
                              w-engagement-utilisateur
                          LET j := j + 1
                    ENDWHILE
                    MODIFIES montage-moteur
                ENDFOR montage-moteur
            ELSE
                OPENADDS montage-moteur
                LET id-mot-dmc OF montage-moteur := nom-objet-m OF
                    plan-piece-7-a-faire
                LET nom-ligne OF montage-moteur := nom-ligne OF
                    w-g-engagement-utilisateur [ i ] OF
                    w-engagement-utilisateur
                LET j := 1
                WHILE j <= 6
                      LET engagements-mot [ j ] OF montage-moteur :
                          engage-utilisateur [ i ] OF
                          w-g-engagement-utilisateur [ i ] OF
                          w-engagement-utilisateur
                      LET j := j + 1
                ENDWHILE
                ADDS montage-moteur
            ENDIF
        ELSE
            IF l-a OF w-g-engagement-utilisateur [ i ]
               OF w-engagement-utilisateur == "O"
            THEN
                FOR EACH usinage
                 WHERE ( id-pie-dmc OF usinage ==
                         nom-objet-m OF plan-piece-7-a-faire ) .AND.
                       ( nom-ligne OF usinage ==
```

```
                                    nom-ligne OF w-a-engagement-utilisateur [
                                      OF w-engagement-utilisateur )
                                BEGINMODIFIES usinage
                                LET i := 1
                                WHILE i <= 6
                                    LET engagements-pie [ i ] OF usinage :=
                                        engagements-pie [ i ] OF usinage +
                                        engage-utilisateur [ i ] OF
                                        w-a-engagement-utilisateur [ i ] OF
                                        w-engagement-utilisateur
                                    LET i := i + 1
                                ENDWHILE
                                MODIFIES usinage
                            ENDFOR usinage
                        ELSE
                            OPENADDS usinage
                            LET id-pie-dmc OF usinage := nom-objet-m OF
                                plan-piece-7-a-faire
                            LET nom-ligne OF usinage := nom-ligne OF
                                w-a-engagement-utilisateur [ i ] OF
                                w-engagement-utilisateur
                            LET i := 1
                            WHILE i <= 6
                                LET engagements-pie [ i ] OF usinage :=
                                    engage-utilisateur [ i ] OF
                                    w-a-engagement-utilisateur [ i ] OF
                                    w-engagement-utilisateur
                                LET i := i + 1
                            ENDWHILE
                            ADDS usinage
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
            LET i := i + 1
    ENDWHILE
ELSE
    DISPLAY " pas de validation suivant choix utilisateur"
ENDIF
ENDRECEIVES plan-piece-7-a-faire
ENDRECEIVES choix-validation
BREAK BLOCK-GENERATES w-engagement-utilisateur
GENERATES w-engagement-utilisateur
ENDPROC validation :

#-------------------------------------------------------------------------
```